VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

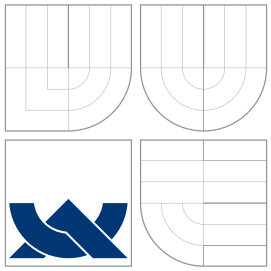
KOMUNIKAČNÍ DESKA PRO ŘÍZENÍ SYSTÉMŮ ŘEZACÍCH STOLŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

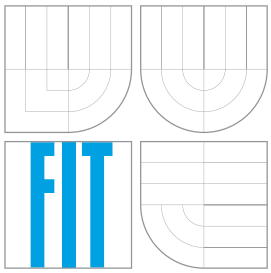
AUTOR PRÁCE
AUTHOR

Bc. ZDENKO BAČÍK

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

KOMUNIKAČNÍ DESKA PRO ŘÍZENÍ SYSTÉMŮ ŘEZACÍCH STOLŮ

COMMUNICATION BOARD FOR OPERATION OF CUTTING TABLES SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ZDENKO BAČÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÁCLAV ŠIMEK

BRNO 2010

Zadání práce

1. Seznamte se s oblastí programovatelných hradlových polí (FPGA). Dle pokynů vedoucího prostudujte konkrétní systém pro syntézu a implementaci obvodů do FPGA čipů.

2. Prostudujte působy přenosu dat mezi CPU a adaptérem periferního zařízení, přičemž se zaměřte na sběrnici PCI.

3. Navrhněte schéma zapojení komunikačního adaptéru do PCI sběrnice, který zabezpečí komunikaci se servo motory jednotlivých os řezacího stroje dle specifikace.

4. Proveďte realizaci adaptéru v podobě desky plošných spojů s využitím vhodného návrhového software.

5. V jazyce VHDL implementujte moduly pro komunikaci po sběrnici PCI a řízení připojených servo motorů.

6. Naprogramujte potřebné ovladače a obslužný firmware, což umožní provozovat kartu pod OS Windows.

7. Vhodným způsobem ověřte funkčnost vytvořené karty a zvažte možnosti dalšího rozšíření.

Abstrakt

Táto práce sa zaoberá problematikou realizácie PCI radiča pomocou technológie FPGA. Popisuje návrh a implementáciu komunikačnej karty určenej do PCI zbernice, ktorá slúži k ovládaniu servomotorov rezacích strojov.

V tomto projekte sú popísané postupy pri návrhu a realizácii hardvérových a softvérových časti komunikačnej karty. Výsledkom práce je fyzicky zhotovené zariadenie, ktoré bude zaradené do procesu výroby.

Abstract

The thesis deals with the issue of implementing a PCI interface controller utilizing the FPGA technology. It describes the design and the implementation of a PCI communication card, which is used to control servomotors in cutting machines.

In the thesis, the steps taken in designing and implementation of hardware and software parts of the communication card are discussed. The result of the thesis is a functional piece of equipment, which is to be manufactured.

Klíčová slova

rekonfigurovateľné obvody FPGA, Spartan II, jazyk VHDL, zbernica PCI, Eagle, radič PCI, ModelSim, ISE

Keywords

reconfigurable circuits FPGA, Spartan II, VHDL language, PCI bus, Eagle, PCI controller, ModelSim, ISE

Citace

Zdenko Bačík: Komunikační deska pro řízení systémů
řezacích stolů, diplomová práce, Brno, FIT VUT v Brně, 2010

Komunikační deska pro řízení systémů řezacích stolů

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Ing. Václava Šimka

.....
Zdenko Bačík
24. května 2010

Poděkování

Týmto by som chcel poďakovať konzultantovi Ing. Václavu Šimkovi za pomoc pri návrhu a realizácii projektu. Ďalej veľká vďaka patrí Ing. Jozefovi Čiernemu, ktorý mi umožnil tento projekt realizovať, financovať a poskytol odbornú pomoc pri jeho realizácii.

Obsah

Kapitola 1

Úvod

V súčasnej dobe počítače tvoria neoddeliteľnú súčasť nášho každodenného života a zasahujú takmer do každého odvetia. Veľké uplatnenie nachádzajú najmä v priemyselnej oblasti, kde ich výpočtová sily umožňuje dosahovať také výsledky, ktoré nie je možné dosiahnuť ľudskou rukou. Počítače sa tak stali súčasťou zariadení, kde vyhodnocujú požiadavky užívateľa a menia ich na potrebné úkony. Ich hlavnou úlohou je tak v hierarchii systému vytvárať komunikačný a riadiaci prvok, ktorý spracúva požiadavky užívateľov pomocou jedného typu rozhrania a mení ich na iný typ rozhrania. Túto úlohu bude zastrešovať aj navrhovaná komunikačná karta, ktorá je predmetom tejto práce.

Komunikačná karta bude predstavovať komplexné zariadenie, ktoré bude vyhodnocovať žiadosti počítačového softvéru a meniť ich na rozhranie, ktorým budú ovládané servomotory rezacích stolov. Komunikačné rozhranie zo strany počítača bude realizované PCI zbernicou. Tento typ rozhrania je zvolený z toho dôvodu, že je súčasťou takmer každého moderného počítača a vyznačuje sa vysokou priepustnosťou dát za cenu pomerne jednoduchej implementácie. Zo strany rezacích strojov sa bude jednať o štandardné rozhranie RS485, ktoré je charakteristické vysokou odolnosťou voči rušeniu a v priemysle má pomerne časté zastúpenie.

Práca sa zaoberá komplexným návrhom a riešením programového a hardvérového vybavenia pre realizáciu komunikačnej karty. Je rozdelená do niekoľkých témetických celkov s logickou nadstavbou.

Úvod práce popisuje teoretickú časť projektu a sú v nej podrobné popísane technológie, ktoré sa využijú pri realizácii komunikačnej karty. Táto časť práce predstavuje pokračovanie pôvodného semestrálneho projektu.

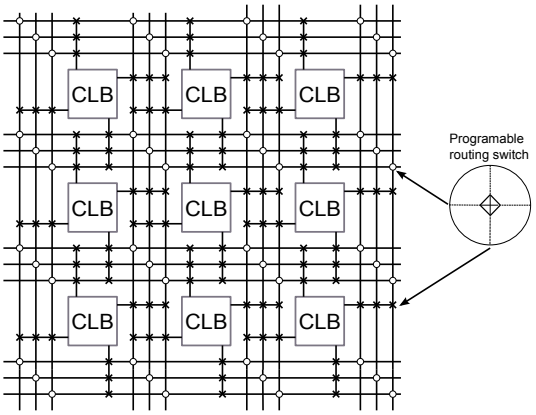
V jadre práce je popísaný proces návrhu, implementácie a fyzickej realizácie komunikačnej karty. Rozdeľuje sa na dve základné častí, kde prvá časť popisuje návrh a realizáciu dosky plošného spoja. Druhá časť popisuje softvérová riešenie radiča PCI zbernice a obslužnej aplikácie.

V závere práce je uvedený proces testovania vyrobeného zariadenia.

Kapitola 2

Programovateľné hradlové polia

Programovateľné hradlové polia predstavujú zariadenia, ktorých funkcia je definovaná užívateľským programom. Obecne sú to štruktúry logických buniek (každá z buniek môže byť nezávisle nakonfigurovaná) v dvojdimenzionálnom poli prepojené pomocou prepínačov (viz obrázok 2.1). Variabilnú funkciu obvodov je tak možné dosiahnuť naprogramovaním logických buniek a ich vzájomným prepojením. Vlastnosti logických buniek sú odlišné v závislosti na výrobcovi obvodu. Jedná o kombináciu niekoľkých vstupov a výstupov. Podrobný popis štruktúry logickej bunky bude predmetom nasledujúceho výkladu (viz kapitola 2.1.1).



Obrázek 2.1: Architektura FPGA obvodu [18].

Hlavnou výhodou FPGA obvodov je, že ich funkčnosť určuje konfiguračný reťazec, ktorý je uložený v konfiguračnej pamäti obvodu. Typicky sa jedna o programovú štruktúru, ktorá je nahrávaná z externej pamäte do pamäte FPGA obvodu počas inicializácie obvodu (viz kapitola 2.1.2). Na rozdiel od procesorov, kde sa program vykonáva pomocou inštrukcii v pamäti a vnútorná inštrukčná sada je pevne daná, programovaním FPGA obvodov sa zmení celá jeho vnútorná štruktúra, a tak konfigurácia FPGA obvodov predstavuje hranicu medzi konfigurovaním hardvéru a softvéru. Tým je možné konštruovať výkonne zariadenia.

2.1 Spartan II

Rodina obvodov Spartan II bola uvedená na trh v roku 2003 firmou Xilinx. Tvorí ju 6 typov obvodov, z ktorých najvýkonnejší obvod môže mať až 200 000 systémových hradiel

a dosahovať maximálnu frekvencie 200 MHz. Udávaná frekvencia je iba teoretická hodnota a v reálnych podmienkach sú tieto hodnoty častokrát nižšie. Frekvencia je značne ovplyvnená spôsobom návrhu aplikácie. Spartan II podporuje až 16 druhov vstupno-výstupných štandardov, čo umožňuje ich široké uplatnenie v rôznych aplikáciách.

Nevýhodou je, že táto rodina obvodov umožňuje iba statickú rekonfiguráciu a dosahovaná reálna frekvencia je pomerne nízka a to pri trende dnešných aplikácii nie je postačujúce. Preto sú tieto obvody postupne nahradzované inými výkonnejšími FPGA obvodmi ako napr. Spartan V, Virtex a Virtex II.

2.1.1 Architektúra obvodu

Rodina obvodov Spartan II je tvorená z nasledujúcich základných blokov (viz obrázok 2.2).

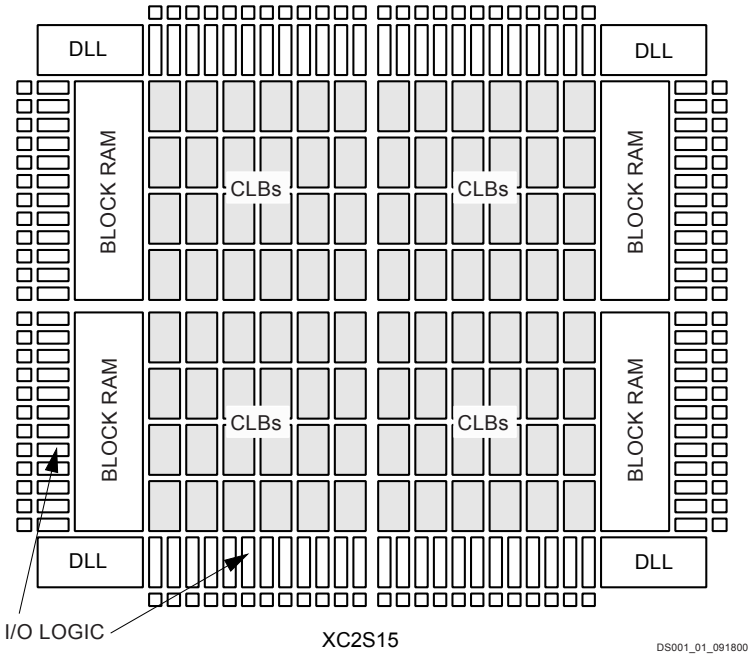
CLB - konfigurovateľný logický blok

IOB - vstupno-výsutpný blok

DLL - distribúcia hodín

RAM - bloková RAM pamäť

GRM - globálna prepojovacia sieť



Obrázek 2.2: Architektura FPGA obvodu Spartan II [18].

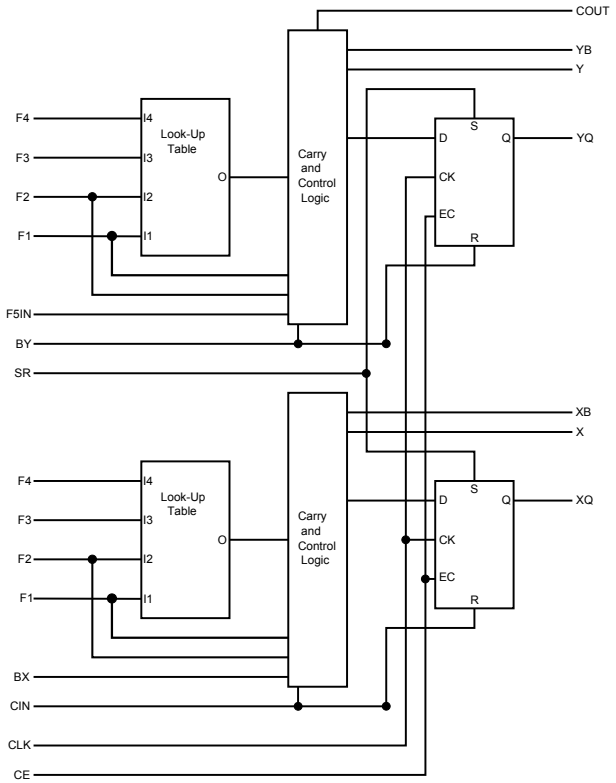
Konfigurovateľný logický blok (CLB)

Konfigurovateľný logický blok obvodu Sparatn II je tvorený štyrmi logickými bunkami, ktoré sú organizované do dvoch *slice* blokov. Každý *slice* blok je tvorený dvoma *looku-up* tabuľkami, *logikou carry prenosu* a dvoma *latch registrami* (viz obrázok 2.3).

Look-up tabuľka (LUT) plní najčastejšie obecnú binárnu funkciu, ktorá ma štyri vstupy a jeden výstup. Kombináciou viacerých LUT tabuliek môžeme vytvárať zložitejšie funkcie. LUT tabuľka môže zároveň vytvárať RAM pamäť o veľkosti 16x1 bit. Vzájomným prepojením viacerých LUT tabuliek je možné dosiahnuť rôzne veľkosti pamätí. Faktom však je, že konštruovať väčšie pamäte pomocou LUT tabuliek nie je efektívne a vedie k plytvaniu systémových prostriedkov.

Carry logika sa využíva pri konštrukcii rýchlych sčítačiek, čítačov a komparátorov. Konfiguračný blok podporuje dva prenosové reťazce *carry* logik (jedna na *slice* blok). Prenosová veľkosť *carry* logiky tým činí 2 bity na jeden konfiguračný blok.

Latch register slúži k synchronizovaniu logiky *slice* bloku s hodinovým signálom FPGA obvodu. Ak je vstupná hodnota registra zmenená, zmena sa na výstupe prejaví na príslušnú hranu hodinového signálu.



Obrázek 2.3: Architektura slice [18].

Súčasťou konfiguračných blokov FPGA sú dva typy multiplexorov F5 a F6, pomocou ktorých je možné kombinovať rôzne *look-up* tabuľky a vytvárať tak funkcie o viac premenných. Je možné dosiahnuť funkcie až o 19-tich premenných.

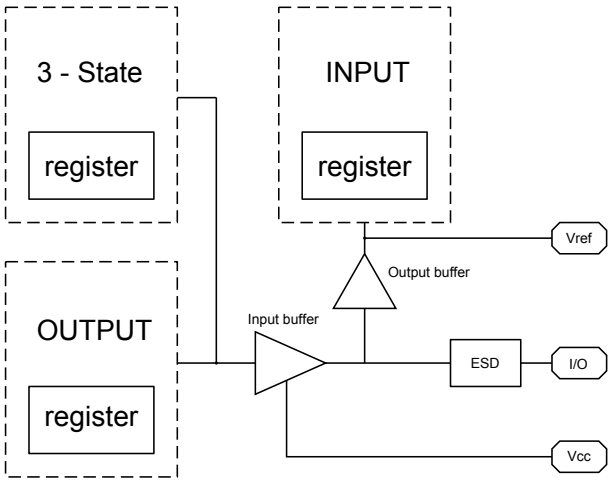
Vstupno-výstpný blok (IOB)

S narastajúcou systémovou frekvenciou obvodov sa značne začali prejavovať oneskorenia v komunikácii medzi zariadeniami *(chip-to-chip delay)*. Toto vo vývoji viedlo k vzniku nových vstupno-výstupných štandardov. Vznik týchto nových štandardov je hlavným dôvodom zavedenia vstupno-výstupných blokov (IOB). Každý vstupno-výstupný blok obvodu Spartan II podporuje až 16 druhov štandardov (viz [18]).

IOB obvod je tvorený troma registrami (viz obrázok 2.4), ktoré umožňujú softwarovo ovládať piny obvodu. Blok ESD zabezpečuje ochranu pred elektrostatickým výbojom. Každý IOB obvod môže byť softwarovo zmenený z výstupného na vstupný a naopak, čo umožňuje prepojenie FPGA obvodov so zdieľanými perifériami.

Vstupná cesta IOB obvodu: Rozlišujeme dva typy ciest. Priama cesta do internej logiky alebo nepriama cesta cez register. Použitím nepriamej cesty je možné využiť register k oneskoreniu signálu a zabezpečiť tak *pad-to-pad time*[1]. Vstupný buffer môže byť nakonfigurovaný na všetky podporované druhy štandardov. Niektoré zo štandardov vyžadujú nastavenie referenčného prahového napätia *Vref*. Každý vstup umožňuje softvérovo nastaviť *pull-up* alebo *pull-down* rezistor.

Výstupná cesta IOB obvodu: Výstupná cesta podporuje 3-stavovú logiku (H, L, Z). Rovnako ako u vstupnej cesty, aj tu rozlišujeme dva typy ciest. Priama cesta z internej logiky alebo nepriama cesta cez register. Každý výstupný buffer môže byť nakonfigurovaný na všetky podporované štandardy. Výber štandardu závisí na hodnote napájacieho napätia *(Vcco)*.



Obrázek 2.4: Vstupno-výstupny blok.

Bloková pamäť (RAM)

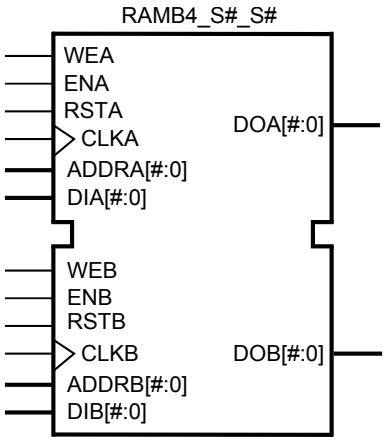
Spartan II obsahuje niekoľko rozsiahlých blokov RAM pamäte. Každý blok je reprezentovaný pomocou bunky o veľkosti 4096 bitov a môže byť nakonfigurovaný na rôzne dátové šírky. Pamäť je vhodná pre uloženie väčšieho objemu dát, konštrukcie FIFO pamäti a FSM

[1]Pad-to-pad time predstavuje čas, ktorý je potrebné budiť logickú hodnotu na vstupe, aby ju obvod zachytil.

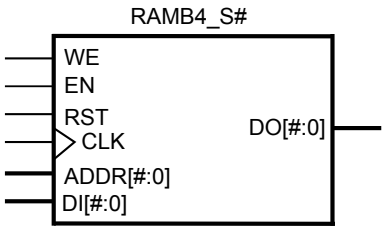
obvodov. Je navrhnutá tak, aby bolo možné konštruovať väčšie pamäťové štruktúry s minimálnymi oneskoreniami medzi jednotlivými blokmi. Pamäť môže byť organizovaná ako jedno-portová a dvoj-portová varianta (viz obrázky 2.5, 2.6) a podporuje dva operačné módy:

- **Čítanie:** v mode čítania je adresa vystavená na hranu hodinového signálu a dáta sa na výstupe objavia za dobu prístupu do RAM pamäte (viz [18]).
- **Zápis:** v mode zápisu je adresa zaregistrovaná na hranu hodinového signálu a dáta sú zapísané do pamäte. Súčasne sa platné dáta objavia aj na výstupe RAM pamäte.

U dvoj-portovej varianty môže nastať konflikt, ak oba porty pristupujú k jednej pamäťovej bunke. Konflikt spôsobuje nekonzistenciu dát a je ho potrebné softvérovo ošetriť. Pri využívaní pamäte je potrebné dodržať správnu jazykovú konštrukciu, aby pamäť bola rozpoznaná v syntéze a zabránilo sa tak zbytočnému plytvaniu prostriedkov.



Obrázek 2.5: Dvojportová bloková pamäť [18].



Obrázek 2.6: Jednoportová bloková pamäť [18].

Prepojovacia sieť

Prepojovacia sieť predstavuje spojovacie cesty medzi prvkami FPGA obvodu. Tvoria ju štyri druhy prepojovacích liniek.

- *Local routing* - Prepojovacia cesta medzi *look-up* tabuľkami a registrami s GRM (globálna prepojovacia matica). Jedná sa o priame prepojenie medzi priľahlými konfiguračnými blokmi, kde sa eliminujú oneskorenia, ktoré vznikajú v GRM.

- *General purpose routing* - Horizontálna a vertikálna prepojovacia cesta pridružená s riadkami a sĺpcami CLB.
- *I/O routing* - Prepojenie ľubovoľných pinov s vnútornou logikou obvodu.
- *Global routing* - Primárna cesta je tvorená sieťou štyroch hodinových signálov, sekundárna cesta je tvorená 24 linkami pre všeobecne použitie.

2.1.2 Konfigurácia

Konfigurácia je proces, kde konfiguračný reťazec je nahrávaný priamo do pamäte FPGA obvodu. Proces konfigurácie prebieha v štyroch fázach a je riadený pomocou signálov INIT, PROGRAM a DONE:

Inicializacia: Zariadenie môže byt inicializované dvoma spôsobmi. Aplikovaním napájacieho napätia alebo nastavením vstupného pinu PROGRAM na aktívnu hodnotu L. Beh konfiguračného procesu je indikovaný signálom DONE. Po úspešnej inicializácii zariadenie prechádza do režimu vymazanie pamäte.

Vymazanie: Zariadenie indikuje proces vymazávania pamäte signálom INIT nastavením na hodnotu L. Podržaním signálov PROGRAM a INIT v aktívnej úrovni L je možne zariadenie podržať v režime vymazávania pamäte. Ukončenie tohto režimu je indikované nadstavením signálu INIT na hodnotu H.

Nahrávanie: V režime nahrávania je konfiguračný reťazec nahrávaný do pamäte zariadenia. Počas nahrávania tohto reťazca FPGA obvod počíta CRC hodnotu [2]. Ak táto hodnota nie je zhodná s referenčnou hodnotou, obvod nahlási chybu nastavením pinu INIT na hodnotu L a zruší konfiguráciu.

Štart aplikácie: Štartovacia sekvencia ukončí konfiguráciu obvodu a spustí aplikáciu.

Rodina obvodov Spartan II podporuje niekoľko režimov konfigurácie. Jednotlivé mody sú vybrané nadstavením pinov M0, M1, M2 (viz [18]).

- Slave sériový mod
- Master sériový mod
- Slave paralelný mod
- Boundary-scan mod

Master sériový mod: V režime master FPGA obvod kontroluje proces konfigurácie generovaním vlastného hodinového signálu, ktorý môže byť v rozmedzí 4 - 60 MHz. Vstupné dáta sú privádzané z externej pamäte na vstupný pin DIN obvodu FPGA. Pri inicializácii zariadenia je frekvencia hodinového signálu 2.5 MHz. Táto frekvencia môže byť zmenená pomocou počiatočnej konfigurácie. V tomto režime musia byť piny M0, M1, M2 nastavené na hodnotu L.

[2] CRC (cyklický redundantní súčet je špeciálna hašovacia funkcia, používaná k detekcii chýb behom prenosu alebo ukladania dát).

Slave sériový mod: V režime slave je proces konfigurácie riadený externým hodinovým signálom. Toto dovoľuje konfigurovať obvod FPGA mikroprocesorom alebo iným FPGA obvodom. Vstupné dáta sú privádzané na vstupný pin DIN krátky okamih pred nábežnou hranou externého hodinového signálu.

Boundary-scan mod: Tento mod sa využíva pri ladení aplikácie FPGA obvodu pomocou JTAG[3] zariadenia. Využíva špeciálne CFGIN inštrukcie.

Slave paralelný mod: Paralelný mod je najrýchlejší zo všetkých konfiguračných režimov. BUSY príznak kontroluje dátový tok a frekvencia hodín počas prenosu 50 MHz.

2.2 VHDL

S narastajúcou zložitosťou obvodov sa grafická reprezentácia logických obvodov stala neprehľadná a zdrojom množstva chýb. To viedlo vývoj k vzniku inej návrhovej techniky k vzniku HDL jazykov. HDL *(Hardware Descripiton Language)* je množina programovacích jazykov, ktorými je možné popísať chovanie elektronických obvodov, konkrétne číslicových obvodov. Pri návrhu pomocou HDL jazykov je možné daný obvod simulovať, čo prispieva k urýchleniu vývoja a minimalizácie chýb.

V praxi sa využívajú najmä dva jazyky z tejto rodiny - VHDL a Verilog. VHDL dominuje v Európe a Verilog v USA. Extrémny nárast počtu hradiel vedie k zavedeniu nových abstraktnejších jazykov rodiny HDL - jazykov SystemC, HandelC, ImpulseC a ďalších, čo prináša výrazné urýchlenie vývoja nových zariadení [12]. Názov VHDL pochádza z akronymu VHSIC *(Very High Speed Integrated Circuit Hardware Description Language)*. Pôvodne bol vyvinutý pre vojenské účely k špecifikácií číslicových systémov, avšak v roku 1987 bol uznaný IEEE štandardom (IEEE Standard 1076-1987) a začal sa používať v praxi. Jazyk nie je zviazaný so žiadnou cieľovou technológiu a má syntax podobnú jazyku PASCAL.

Obecné programovacie jazyky sú založené na sekvenčnom modely, avšak digitálne číslicové obvody vykonávajú operácie paralelne. Pre tento dôvod obecné programovacie jazyky nemôžu presne popísať alebo modelovať chovanie číslicových obvodov. VHDL je tak navrhnutý k modelovaniu paralelných operácii. Obecne v HDL jazykoch rozlišujeme premenné a signály. Hodnoty premenných sú zmenené bez oneskorenia a hodnoty signálov sú zmenené s malým alebo definovaným oneskorením. Použitím signálov je možné dosiahnuť simuláciu pracujúcu ako hardvér.

2.2.1 Popis číslicoveho obvodu jazykom VHDL

Jednotlivé časti zariadenia popísane jazykom VHDL môže byť rozdelené na niekoľko nezávislých (paralelných) pracujúcich častí. Tieto časti sa vo VHDL volajú komponenty. Komponenta sa dá popísať buď štrukturálne alebo behaviorálne.

Štrukturalny popis

Štrukturálny popis popisuje funkčnosť obvodu pomocou komponent a ich prepojením pomocou signálov. Komponenty môžu byť dekomponované na subkomponenty a samostatne popísané užívateľom alebo knižnicami dodávaných výrobcom. Tento popis má výhodu v

[3] JTAG - Joint Test Action Group je štandard založený na architekture Boundary Scan.

tom, že je veľmi blízky finálnej obvodovej realizácii a návrhár má do istej miery pod kontrolou proces syntézy a časovania. Štrukturálny popis môže mať viac úrovní hierarchie, kde každá komponenta môže byť opäť popísaná na úrovni štruktúry.

Behaviorálny popis

Komponenty na najnižšej úrovni sú vždy popísane behaviorálne. Tento popis sa využíva hlavne pri simulácii. Behaviorálny popis využíva fakt, že chovanie obvodu môžeme popísať algoritmom. Pri tomto popise sa používajú konštrukcie bežné v programovacích jazykoch (cykly, procedúry, funkcie atd.). V popise nie sú brané do úvahy obvodové detaily a tvorba zapojenia sa necháva na proces syntézy. Nie vždy musí byť jazyková konštrukcia obvodovo realizovateľná (napr. rekurzia).

2.2.2 Proces syntézy

Syntéza je proces transformácie medzi rôznymi druhmi popisu. Cieľom syntézy je vylepšiť rôzne parametre, napríklad rýchlosť, spotreba, testovateľnosť atd. Proces syntézy je rozdelený do niekoľkých krokov.

Behaviorálna syntéza

V behaviorálnej syntéze je daný algoritmus popísaný na najvyššej úrovni (sčítačky, posuvný register, pamäť, riadiaca logika). Popis je sústredený na obmedzenia vzťahujúce sa na vstup, výstup a užívateľské obmedzenia. Výstupom behaviorálnej syntézy je popis na úrovni medziregistrových prenosov, ktoré majú rovnaké chovanie ako vstupný obvod. Nevýhodou je, že optimalizácia je riadená nástrojmi realizujúcich syntézu. Jedinou možnosťou ako riadiť výsledok syntézy je nadstavenie obmedzení [13].

RTL syntéza

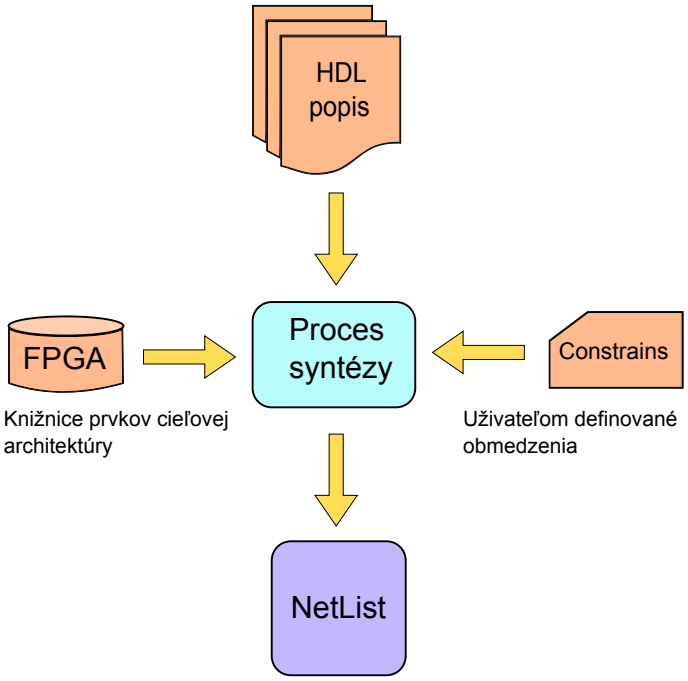
RTL syntéza je syntéza na úrovni medziregistrových prenosov. Vstupný obvod je popísaný pomocou registrov, čítačov a automatov, ktoré sú prevedené do booleovskej logiky (primitívne hradla typu AND, OR, flip-flop). Táto syntéza je významná tým, že oddeľuje dátovú a riadiacu cestu. Riadiacu cestu najčastejšie tvorí FSM automat. Dátovú cestu tvoria rôzne logické siete, sčítačky, registre atd. Celá dátová cesta je tak riadená pomocou signálov z riadiacej cesty. Syntetizátor z tohto popisu vygeneruje popis na úrovni hradiel. Výsledok syntézy obsahuje optimalizovanú cestu, pamäť a riadiace štruktúry [13].

Logická syntéza

Logická syntéza je syntéza na úrovni hradiel a medziregistrových prenosov. Z HDL popisov je vytvorený NetList prvkov cieľovej technológie.

Proces syntézy pracuje s podmnožinou jazyka VHDL - syntetizovatelné VHDL. Z toho dôvodu bol definovaný štandadrd IEEE 1076.6 - 1999 pre RT syntézu. Štandard definuje podmnožinu VHDL konštrukcií, ktorá je podporovaná všetkými nástrojmi syntézy.

Na obrázku 2.7 je znázornený proces syntézy, pomocou ktorého je popis konštrukcií jazyka prevedený na cieľovú architektúru FPGA obvodu.



Obrázek 2.7: Syntéza [12].

Prvý krok syntézy je nezávislý na cieľovej architektúre a je spoločný pre všetky typy zariadení. Vstupom je tak popis v HDL jazyku, knižnice prvkov cieľovej technológie a užívateľom definované obmedzenia. V ďalšom kroku dochádza k mapovaniu prvkov na cieľovú architektúru a vytvára sa tak optimalizovaný NetList cieľovej technológie. V priebehu syntézy sa prevádzajú optimalizácie na rýchlosť, spotrebu a testovateľnosť.

Kapitola 3

PCI

Zbernica PCI *(Peripheral Component Interconnect)* predstavuje štandard pre pripojenie periférnych zariadení v rámci architektúry PC. História vývoja PCI siaha už do roku 1993, kde sa objavuje ako náhrada za ISA zbernicu a je vyvinutá najmä pre grafické operačné systémy Window a OS/2. Prvá špecifikácia zbernice pracuje na frekvencii 33 Mhz a neskôr je tento štandard rozšírený o dvojnásobok frekvencie na 66 Mhz. PCI bola prvýkrát použitá v osobných počítačoch v roku 1992 firmou Intel na matičnej doske *Alfredo* pre procesor 486. Jednalo sa vtedy o štandard PCI verzie 1.0. V súčasnosti posledným platným štandardom je PCI verzia 3.0 z roku 2004. Cieľom tejto kapitoly je popísať architektúru PCI a základy komunikačného protokolu [14].

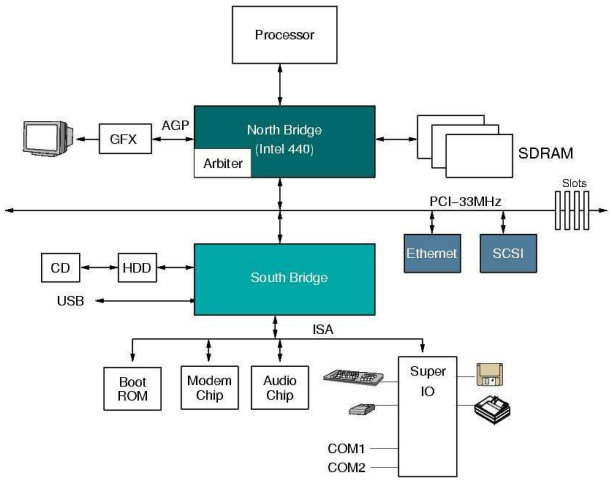
3.1 Architektúra založená na PCI zbernici

Model architektúry, založenej na PCI zbernici, je znázornený na obrázku 3.1. Z dôvodu rôznych komunikačných rýchlostí je architektúra rozdelená na *north bridge* (severný most) a *south brdige* (južný most). *North bridge* realizuje rýchlu komunikáciu medzi procesorom, systémovou zbernicou a grafickou pamäťou. Súčasťou *north bridge* je aj zbernica PCI, ktorá slúži na pripojenie pomalších zariadení ako sú napríklad sieťový adaptér, SCSI radič atd. *South bridge* realizuje pomalšiu komunikáciu a predstavuje tak rozhranie medzi pomalými perifériami (klávesnica, myš) a PCI zbernicou. Zariadenie pripojené na *south bridge* sa chová ako PCI zariadenie.

S príchodom nových a rýchlejších zberníc sa architektúra personálnych počítačov postupne menila. PCI zbernica sa stala súčasťou *south bridge* a je nahradená novou a rýchlejšou zbernicou PCI-Express.

3.2 Komunikačný protokol

Každá operácia na PCI zbernici musí mať dvoch účastníkov a komunikácia medzi nimi musí dodržiavať presne stanovený protokol. Zariadenie, ktoré zakladá operáciu, sa volá iniciátor *(master)* a zariadenie, s ktorým je komunikácia nadviazaná, sa volá *target* (cieľové zariadenie). Komunikácie medzi týmito zariadeniami prebieha v niekoľkých krokov (viz kapitola 3.2.1) a najčastejšie pomocou blokového prenosu (viz kapitola 3.2.2).



Obrázek 3.1: Architektúra založená na PCI zbernici [11].

3.2.1 Priebeh komunikácie

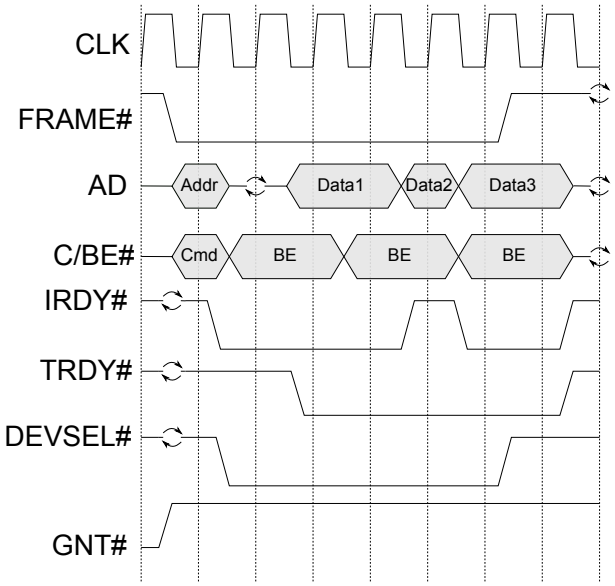
Komunikácia na zbernici prebieha pomocou niekoľkých krokov, kde vykonanie týchto krokov predstavuje transakciu 3.2. Počas prenosu môže nastať niekoľko transakcii za sebou. Každá transakcia má svoj začiatok a koniec. Zbernica PCI je zdieľaná, adresa a dáta sú multiplexované na rovnaké signály. Dosiahne sa tak výrazná redukcia počtu vodičov na zbernici.

1. Iniciátor nadstaví signál FRAME# na aktívnu hodnotu L a súčasne na zbernici vystaví adresu zariadenia, s ktorým chce komunikovať. Spolu s adresou sa posiela príkaz *(command)*, ktorý špecifikuje typ transakcie (viz kapitola 3.6). Ak sa jedná o konfiguračné prenosy, namiesto adresy je využívaný signál IDSEL, ktorým sa vyberá konkrétne zariadenie na zbernici. Konfiguračný prenos sa využíva k prístupom do konfiguračnej pamäte.

2. Ak cieľové zariadenie rozpozná svoju adresu na zbernici (alebo zachytí aktívny signál IDSEL), aktivuje signál DEVSEL# (L). Aktivovaním signálu TRDY# (L) sa určí, že cieľové zariadenie vystavilo platné dáta na zbernicu a zahájila sa tak prvá dátová fáza. Signál DEVSEL# je aktívny počas celej doby trvania transakcie.

3. Nasleduje prenos dát po zbernici. Táto komunikácia je synchronizovaná pomocou signálov IRDY# (zo strany iniciátora) a TRDY# (zo strany cieľového zariadenia). Tieto signály sú využité k riadeniu komunikácie medzi zariadeniami (vkladanie čakacích cyklov), ktoré pracujú na rôznych prenosových rýchlostiach. Ak je jeden zo signálov deaktivovaný, príslušná transakcia je pozastavená a čaká sa maximálne po dobu osem hodinových cyklov, kým nie je signál opäť aktívny. Ak signál nie je aktivovaný ani na ôsmy hodinový cyklus, komunikácia na zbernici sa ukončí.

4. Cieľové zariadenie nikdy nevie, koľko dát sa bude prenášať. Koniec prenosu je určený deaktivovaním signálu FRAME# (H) a aktivovaním signálu IRDY# (L) v priebehu poslednej dátovej fáze. O ukončenie transakcie môže požiadať aj cieľové zariadenie aktivovaním signálu STOP# (L) viz. 3.10.



Obrázek 3.2: Komunikačný protokol [11].

3.2.2 Blokový prenos

Blokový prenos je tvorený adresnou a dátovou fázou. V adresnej fáze sa prenesie adresa a typ prenosu. V každej nasledujúcej fáze sú prenášané dáta. Z hľadiska maximálneho objemu prenesených dát za jednotku času je tento prenos najvhodnejší. Cieľové zariadenie nikdy nevie, koľko dát sa bude prenášať, preto je potrebné identifikovať koniec prenosu.

3.2.3 Jednoduchý prenos

Jednoduchý prenos je tvorený jednou adresnou a jednou dátovou fázou. V adresnej fáze sa prenesie adresa a typ prenosu. V nasledujúcej fáze sú prenesené dáta a transakcia je ukončená. Ak iniciátor vykonáva blokový prenos (v súčasnosti väčšina základových dosiek), cieľové zariadenie musí požiadať o jednoduchý prenos aktivovaním signálu STOP#.

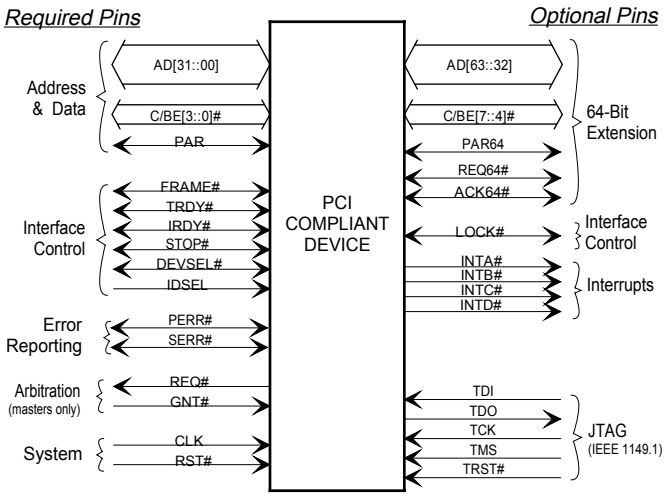
3.3 Definicia signálov

Rozhranie PCI zbernice vyžaduje minimálne 47 pinov pre cieľové zariadenie a 49 pinov pre iniciátora. Obrázok 3.3 zobrazuje signály zoskupené vo funkčných skupinách. Povinné signály sú na ľavej strane a voliteľné na pravej strane. Smerová šípka určuje smer komunikácie medzi iniciátorom a cieľovým zariadením.

3.3.1 Systémové signály

CLK

CLK *(clock)* predstavuje hodinový signál, ktorý zabezpečuje časovanie transakcií na PCI zbernici. Všetky signály na PCI zbernici sú vzorkované na nábežnú hranu hodinového signálu CLK. Frekvencia CLK signálu môže byť v intervale 0 až 33 MHz (špec. 2.0) a 0 až 66 MHz (špec. 2.1). PCI zariadenia musia pracovať v týchto intervaloch. Frekvenciu hodín je



Obrázek 3.3: PCI pin list [7].

možné zmeniť kedykoľvek počas operácie. Výnimku tvoria zariadenia, ktoré sú integrované na matičnej doske. Tieto zariadenia musia mať fixnú frekvenciu 33 MHz. Hodinový signál je možné zastaviť iba v hodnote L.

RST#

Reset *(reset)* je signál využívaný k reštartovaniu PCI zbernice. Signál je asynchrónny a jeho aktivovaním (L) sú všetky signály na zbernici nadstavené do počiatočného (inicializačného) stavu.

3.3.2 Adresové a dátové signály

AD[31:00]

Adresa a dáta sú multiplexované na rovnaké signálové cesty. Počas adresnej fázy sa fyzická adresa rozkladá v celom dvojslove. Počas dátovej fázy AD[31] predstavuje najviac významový bit (MSB) a AD[0] najmenej významový bit (LSB).

C/BE[3:0]#

C - *command* (príkaz), BE - *byte enable* (povoľovací príkaz) predstavuje riadiace signály zbernice a sú multiplexované na spoločnú signálovú cestu. Počas adresnej fázy sú signály využité ako definícia typu prenosu (viz. kapitola 3.6) a počas dátovej fázy sa signály využívajú k indikácii počtu platných bytov v práve adresovanom dvojslove.

PAR

Signál parity je zdieľaný signál a slúži k validácii signálov AD[31:00] a C/BE[3:0]. Počas adresnej fázy sa nadstavuje jeden hodinový takt za každou adresnou fázou a je riadený iniciátorom. Počas dátovej fázy sa nadstavuje jeden hodinový takt po nadstavení signálu IRDY# (u zápisových transakcii, riadený iniciátorom) alebo po nastavení signálu TRDY# (u čítacích transakcii, riadený cieľovým zariadením).

3.3.3 Riadiace signály

FRAME#

Signál riadený iniciátorom. Indikuje začiatok transakcie na zbernici. Ak zariadenia na zbernici detekujú aktívny signál FRAME#, porovnajú adresu na zbernici s bázovou adresou a určí sa tak, ktorému zariadeniu patrí daná transakcia. U blokového prenosu je signál aktívny počas celej doby prenosu. Počas jednoduchého prenosu je signál aktívny jeden hodinový takt.

IRDY#

Signál riadený iniciátorom. Indikuje schopnosť iniciátora dokončiť práve začatú dátovú fázu transakcie. Používa sa v spojení so signálom TRDY# (viz 3.3.3). Dátová fáza môže byť dokončená, ak sú oba signály aktívne. Počas zápisu IRDY# indikuje platnosť dát a počas čítania indikuje, že iniciátor je schopný akceptovať dáta.

TRDY#

Signál riadený cieľovým zariadením. Využíva sa v spojení so signálom IRDY# (viz. kapitola 3.3.3). Počas čítania indikuje platnosť dát a počas zápisu indikuje, že cieľové zariadenie je schopné akceptovať dáta. V spojení s IRDY# signálom sa využíva na vkladanie čakacích cyklov pri rôznych rýchlostiach zariadení.

STOP#

Signál riadený cieľovým zariadením. Aktivovaním signálu spoločne so signálom TRDY# cieľové zariadenie žiada o ukončenie vykonávanej transakcie.

LOCK

Signál sa využíva k uzamknutiu adresy cieľového zariadenia počas atomických operácii.

DEVSEL#

Signál riadený cieľovým zariadením. Aktivovaním signálu zariadenie indikuje, že je pripravené k začatiu transakcie. V závislosti od toho, na ktorý hodinový cyklus zariadenie aktivuje signál DEVSEL#, rozlišujeme tri typy zariadení: *fast, medium a slow*. Táto informácia je pevne nastavená výrobcom a uložená v stavovom registre zariadenia.

3.3.4 Signály arbitráže

REQ#

Signál riadený iniciátorom. Využíva sa ako žiadosť o pridelenie zbernice. Každý iniciátor musí mať vlastný signál REQ#.

GNT#

Signál slúži ako odozva na požiadavku signálu REQ#. Jeho nadstavením sa indikuje pridelenie zbernice žiadateľovi.

3.3.5 Signály hlásenia chýb

PERR#

Signál indikuje chybu dátovej parity počas transakcií na zbernici. Aktivuje sa minimálne dva hodinové cykly po dokončení dátovej fázy, keď je detekovaná chyba parity (signál PAR 3.3.2).

SERR#

Signál indikuje chybu adresnej parity, dátovej parity, špeciálnych príkazov a iných systémových chýb.

3.3.6 Signály pre riadenie prerušenia

PCI zariadenie môže vyvolať prerušenie na PCI zbernici pomocou asynchrónnych signálov INTA#, INTB#, INTC#, INTD#. (viz 3.8).

3.4 Pamäťový priestor PCI

PCI štandard podporuje tri typy pamäťových priestorov:

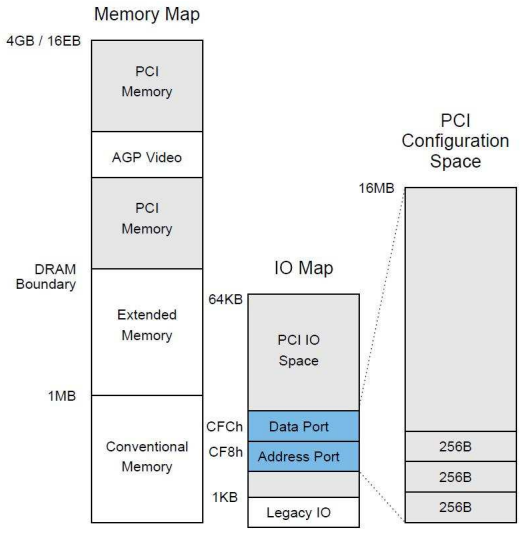
1. **I/O priestor:** Pamäťový priestor určený pre I/O operácie. Operačný systém obvykle využíval tento priestor pre prístup k riadiacim registrom periférnych zariadení. Veľkosť priestoru je vo všeobecnosti 4G (2^{32}). Výnimku tvoria procesory Intel, ktoré sú schopné adresovať I/O operácie maximálne 16 bitmi. Veľkosť ich I/O priestoru je tak 64kB. Niektoré procesory I/O priestor vôbec nepodporujú. V súčasnosti sa prístup k I/O priestoru zachováva len z historického dôvodu.

2. **Pamäťový priestor:** Pamäťový priestor PCI zariadenia je mapovaný do pamäťového priestoru PC. Tento spôsob tak umožňuje pracovať s obsahom dát periférnych zariadení, ako keby bol súčasťou RAM pamäte. Mapovapnie PCI zbernice do pamäťového priestoru závisí na použití architektúry PC. Obvykle tieto systémy majú 32-bitovú alebo 64-bitovú architektúru, čo odpovedá pamäťovému priestoru v rozsahu od 0 - 4GB alebo 0 - 16EB. Typické rozloženie pamäťového priestoru je vidieť na obrázku 3.4.

3. **Konfiguračný priestor:** Konfiguračný priestor je určený pre uloženie konfiguračných dát o jednotlivých PCI zariadeniach. Každé PCI zariadenie má vymedzený priestor 256 bajtov pre uloženie konfiguračných informácií. Prístup do týchto registrov je možný pomocou konfiguračných cyklov 3.5.

3.5 Konfiguračný priestor

Konfiguračný priestor je štruktúra registrov, umiestnená v každom funkčnom PCI zariadení o veľkosti 256 bajtov. Tento priestor je rozdelený do dvoch častí, kde hlavičku predstavuje prvých 64 bajtov a je definovaná špecifikáciou PCI a zvyšných 192 bajtov je použitých k uloženiu špecifických informácií zariadenia. Konfiguračný priestor zariadenia musí byť prístupný v každom čase pomocou konfiguračných cyklov.



Obrázek 3.4: Adresový priestor PCI [11].

3.5.1 Štruktúra konfiguračného priestoru

Obrázok 3.5 zobrazuje formát hlavičky konfiguračného priestoru. Farebne označené registre sú povinné.

Vendor ID

Pole identifikujúce výrobcu zariadenia. Toto číslo je jedinečné a je prideľované autoritou PCI SIG. Hodnota 0xFFFF hexa je nepovolená hodnota vendor ID.

Device ID

Identifikácia zariadenia. Hodnota je určená výrobcom.

Revision ID

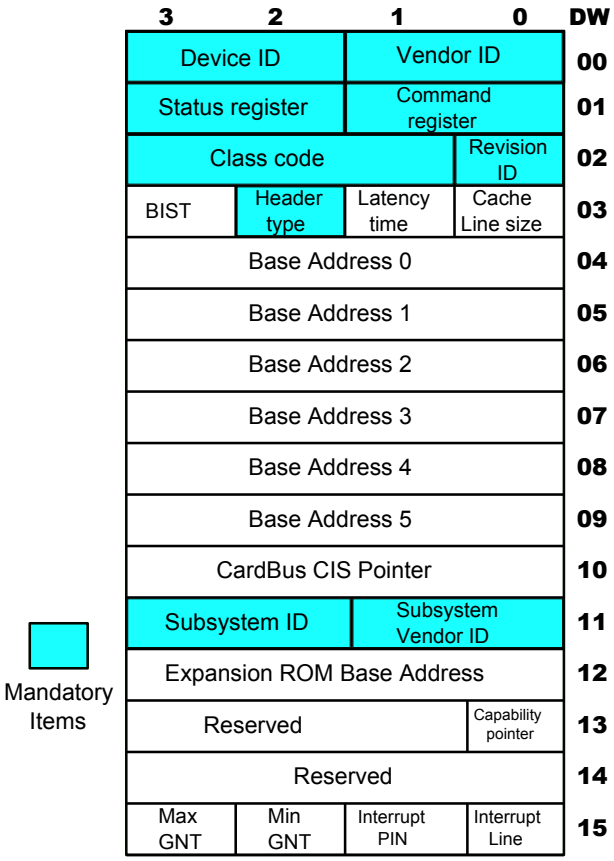
Register špecifikuje revíziu výrobku a určuje ho výrobca. Nula je nepovolená hodnota.

Header type

Register určuje štruktúru zvyšných 192 bitov konfiguračného priestoru. Určuje, o aký typ zariadenia sa jedná. Ak je hodnota 0, jedná sa o jednoduché zariadenie. Ak je hodnota 1, jedná sa o multifunkčné zariadenie.

Class code

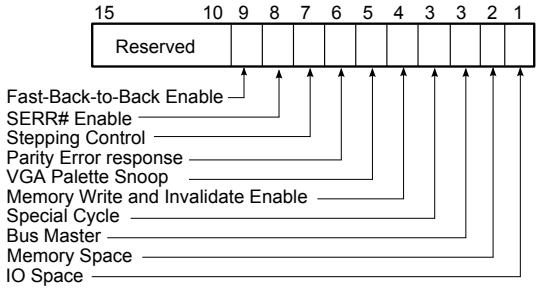
Register určený pre čítanie. Určuje obecné funkcie zariadenia (napr. multimediálne, sieťové atd.). Register je rozdelený na tri časti, kde horných 8 bitov obsahuje základnú funkčnosť zariadenia. Spodné bity špecifikujú presnejšiu funkčnosť zariadenia.



Obrázek 3.5: Konfiguračný priestor PCI [11].

Command register

Príkazový register. Definuje, ktoré operácie PCI zariadenie sú podporované. Zapísaním hodnoty 0 do tohto registru je zariadenie odpojené od PCI zbernice. Túto funkciu musia podporovať všetky zariadenia. Register využíva dolných 9 bitov, kde každý bit nesie určitú funkčnosť. Nie všetky bity musí zariadenie nevyhnutne podporovať. Štruktúru príkazového registru znázorňuje obrázok 3.6 a ich popis znázorňuje tabuľka 3.1.



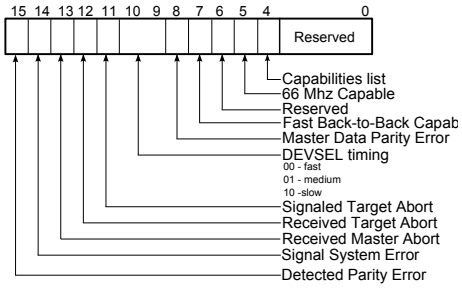
Obrázek 3.6: Štruktúra príkazového registra [7].

Pozícia bitu	Popis
0	Definuje schopnosť zariadenia pristupovať k I/O priestoru. Ak je hodnota bitu nastavená na logickú 0, prístup je zakázaný, inak je prístup povolený. Počiatočná hodnota je logická 0.
1	Definuje schopnosť zariadenia pristupovať k pamäťovému priestoru. Ak je hodnota bitu nastavená na logickú 0, prístup je zakázaný, inak je prístup povolený. Počiatočná hodnota je logická 0.
2	Definuje schopnosť zariadenia vystupovať ako iniciátor na PCI zbernici. Nastavením bitu na hodnotu logickej 0 zakazuje túto schopnosť. Počiatočná hodnota je logická 0.
3	Definuje schopnosť zariadenia používať špeciálne cykly. Ak je bit nadstavený na logickú 1, zariadenie môže využívať špeciálne cykly. Počiatočná hodnota je logická 0.
4	Bit platí len pre iniciátora. Definuje schopnosť iniciátora využívať príkazy zápisu do pamäte a zneplatnenia. Ak je bit nastavený na logickú 1, zariadenie môže generovať príkazy pamäťového prístupu. Ak je bit nadstavený na logickú 0, zariadenie prednostne používa pamäťový zápis. Počiatočná hodnota je logická 0. Pre cieľové zariadnie je hodnota bitu vždy 0.
5	Bit definuje typ prístupu grafických zariadení do VGA paletových registrov.
6	Definuje schopnosť zariadenia - hlásenie chyby parity signálom PERR#. Nastavením bitu do logickej 0 sú signály parity ignorované. Počiatočná hodnota je logická 0.
7	Bit definuje používanie krokovej funkcie. Pomocou tejto funkcie je možné rozdeliť AD linky do niekoľkých skupín. Iniciátor potom postupne vybudí každú skupinu v inom hodinovom takte. Zabráni sa tak prúdovým špičkám, ktoré môžu vzniknúť na zbernici. Zariadenie, ktoré nepoužíva krokovanie, musí mať trvalo nastavený bit v logickej 0. Zariadenie, ktoré využíva krokovanie, musí nastaviť tento bit na hodnotu logickej 0. Bit bude po reštarte ovládaný operačným systémom (napr. pri úspornom režime bude 1 a v režime plného výkonu bude 0).
8	Bit umožnuje zariadeniu generovať signál SERR# na zbernicu. Ak je bit nastavený v logickej 1, zariadenie môže využívať tento signál. Počiatočná hodnota bitu je logická 0.
9	Povoľuje alebo zakazuje využívanie *fast back-to-back* transakcii (prenos dát s rôznymi cieľovými zariadeniami).

Tabulka 3.1: Popis command registru.

Status register

Stavový register sa využíva k ukladaniu udalostí vykonávaných na PCI zbernici. Definíciu jednotlivých bitov je možné vidieť na obrázku 3.7 a ich popis znázorňuje tabuľka 3.2. Zariadenie nemusí nevyhnutne podporovať všetky vlastnosti.



Obrázek 3.7: Štruktúra stavového registra [7].

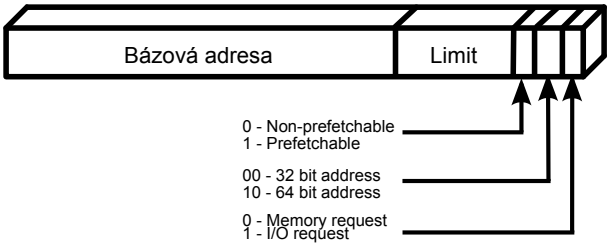
Pozícia bitu	**Popis**
4	Bit iba na čítanie. Určuje, či zariadenie implementuje rozširujúce možnosti. Ak je bit nastavený na hodnotu 1, indikuje, že na adrese 34h je definovaný ukazateľ na rozširujúce možnosti.
5	Bit určuje, na akej frekvencii zariadenie pracuje. Hodnota bitu 0 definuje frekvenciu 33 Mhz.
7	Bit iba na čítanie. Určuje, či zariadenie je schopné akceptovať *back-to-back transakciu.*
8	Bit implementovaný iniciátorom. Využíva sa pri detekcii chyby parity.
9 - 10	Dvojica bitov definuje časovanie signálu DEVSEL#. Bity sú určené na čítanie a indikujú, na ktorý hodinový tak zariadenie aktivuje signál DEVSEL#.
11	Bit implementovaný cieľovým zariadením. Nastavuje sa vždy počas ukončenia transakcie cieľovým zariadením. Nie všetky zariadenia musia využívať tento bit.
12	Bit implementovaný iniciátorom. Nastavný vždy počas ukončenia transakcie cieľovým zariadením. Všetky master zariadenia musia implementovať tento bit.
13	Bit implementovaný iniciátorom. Nastavený počas ukončenia transakcie iniciátorom. Všetky master zariadenia musia implementovať tento bit.
14	Bit nastavený počas aktivovania signálu SERR# . Zariadenia, ktoré nevyužívajú signál SERR#, nemusia implementovať tento bit.
15	Bit nastavený počas detekcie chyby parity.

Tabulka 3.2: Popis status registru.

Register bázovej adresy

Register slúži k alokácii PCI zariadenia do pamäťového priestoru. Určuje koľko, kde a aký typ pamäte sa musí vyhradiť v systéme pre namapovanie zariadenia. Štruktúru registru bázovej adresy je možné vidieť na obrázku. 3.8.

Register je rozdelený na tri časti. Časť *typ požiadavky* má veľkosť 4 bity a je pevne nastavená výrobcom karty. Určuje základné parametre adresového priestoru. Časť *limit* určuje, aký veľký pamäťový priestor má byť pridelený zariadeniu v pamäti. Počet vyhradených bitov pre túto časť určuje, o aký veľký priestor sa jedná. Ak výrobca nadstaví veľkosti



Obrázek 3.8: BAR register.

tejto časti na 10 bitov, tak zariadenie bude požadovať o pridelenie pamäti o veľkosti 1024 bitov (2^{10}). Posledná časť registra *bázová adresa* určuje hodnotu, ktorá odpovedá bázovej adrese prideleného priestoru v systéme. Pridelená hodnota je závislá na type systému. U 32 bitových architektúr nikdy nepresiahne hodnotu 4GB.

Proces alokácie pamäte prebieha v troch krokoch.

1. Systémový softvér (BIOS) zapíše do neinicializovaného BAR registra samé jednotky. Bity, ktoré sú napevno nastavené výrobcom, nebudú ovplyvnené.

2. Systém prečíta hodnotu z BAR registru a určí tak požadovaný typ a množstvo pamäte. Po získaní všetkých požiadaviek (od každého zariadenia na zbernici) systém rozdelí pamäťový priestor.

3. Systémový softvér zapíše hodnoty do BAR registru, ktoré odpovedajú bázovej adrese prideleného priestoru.

4. V poslednom kroku je nastavený príslušný bit v *command registr* (1. bit pre pamäťové operácie) a tým je povolený prístup k cieľovému zariadeniu.

3.6 Príkazy PCI zbernice

Príkazy PCI zbernice indikujú cieľovému zariadeniu, o aký typ transakcie žiada iniciátor. Príkazy zbernice sú zakódované v signáloch C/BE[3:0]# počas adresnej fázy. Tabuľka 3.3 popisuje, aké typy príkazov môžu prebiehať na PCI zbernici.

Použitie pamäťových príkazov závisí na pozícii dát v pamäti a množstva dát, ktoré budú vyčítané. Pamäť je tak možné rozdeliť na dva typy:

1. **prefetchable** - pamäť je možné prečítať do cache pamäte procesoru a pracovať s ňou bez opätovného prístupu k pamätiam zariadenia. Tento typ je možné využiť iba vtedy, ak do pamäte zariadenia zapisuje iba procesor (napr. grafická karta).

2. **non-prefetchable** - do pamäte sa musí vždy pristupovať priamo. Každá pamäťová operácia znamená prístup do pamäte zariadenia, aby procesor pracoval s aktuálnymi dátami. Tento typ pamäte je potrebné využiť vtedy, ak pamäť zariadenia využíva niekoľko procesov.

C/BE[3:0]	Popis
0000	**Potvrdenie prerušenia** - Príkaz využíva systémový interrupt kontrolór. Slúži k indikácii veľkosti vektoru prerušenia.
0001	**Špeciálny cyklus** - Príkaz určuje jednoduchý broadcast mechanizmus na zbernici PCI.
0010	**I/O čítanie** - Príkaz použitý k čítaniu dát z namapovaného I/O adresového priestoru.
0011	**I/O zápis** - Príkaz sa využíva k zápisu dát do namapovaného I/O adresového priestoru.
0100	**Rezervované**
0110	**Čítanie z pamäte** - Príkaz použitý pre čítanie dát z namapovaného pamäťového priestoru.
0111	**Zápis do pamäte** - Príkaz použit7 pre zápis dát do namapovaného pamäťového priestoru.
1000	**Rezervované**
1001	**Rezervované**
1010	**Konfiguračne čítanie** - Príkaz použitý pre čítanie z konfiguračnej pamäte.
1011	**Konfiguračne zápis** - Príkaz využívaný pre zápis do konfiguračnej pamäte. Adresovanie konfiguračného zápisu je rovnaké ako konfiguračné čítanie.
1100	**Viacnásobné pamäťové čítanie** - Príkaz je systematicky rovnaký ako pamäťové čítanie, avšak zahrňuje viacnásobné čítanie cach pamäte pred odpojením zariadenia od zbernice. Pamäťový kontrolór musí zabezpečiť napĺňanie cahe pamäte, pokým je signál FRAME# aktivovaný.
1101	**Dvojitý adresný cyklus** - Príkaz použitý pre adresovanie 64-bitovej adresy.
1110	**Lineárne pamäťové čítanie** - Príkaz je systematický podobný ako pamäťové čítanie, avšak iniciátor týmto príkazom určí vyčítanie obsahu celej cache pamäte.
1111	**Pamäťový zápis a zrušenie platnosti** - Príkaz garantuje zápis minimálne jedného riadku cache pamäte.

Tabulka 3.3: Popis príkazov.

3.7 Arbitráž zbernice

Na zbernicu PCI môže byť pripojených súčasne niekoľko zariadení. Aby nedochádzalo ku kolíziám, je potrebne riešiť prístup zariadení k zbernici. PCI definuje centralizovaný spôsob arbitráže to znamená, že riadiaci obvod je umiestnený v *north bridge* a ku každému zariadeniu sú privedené dva signály *REQ#* a *GNT#*. Zariadenie, ktoré chce získať kontrolu nad zbernicou musí o to požiadať nadstavením signálom REQ# do aktívnej hodnoty L. Na základe požiadavku riadiaci obvod rozhodne o pridelení zbernice aktivovaním signálu GNT. PCI zbernica umožňuje *skrytú arbitráž*, to znamená, že o zbernicu môže žiadať iné zariadenie počas vykonávania transakcie.

3.8 Obsluha prerušenia

Prerušenie na PCI zbernici je asynchrónna udalosť, vyvolaná najčastejšie vonkajším impulzom. Na základe tejto udalosti sa preruší aktuálne vykonávaná udalosť a obslúži sa požiadavka prerušenia. PCI zbernica ma štyri signály pre vyvolanie prerušenia. Radič prerušenia je umiestnený najčastejšie v *south bridge* (závisí na architektúre PC) a jeho úlohou je vyhodnotiť prioritu prerušenia a zaslať požiadavku procesoru o vykonanie prerušovacej rutiny (viz [15]).

3.9 Obsluha chyby

Pre zaistenie kontroly dát na PCI zbernici sa využíva systém detekcie parity. Počas každej transakcie zariadenie kontroluje paritu na základe paritného signálu PAR. Signál je nadstavený tak, aby 36 bitov (AD[31:00], C/BE[3:0]#) obsahovalo párny počet jednotiek. V prípade detekcie chybnej parity zariadenie nadstaví signál PERR# alebo SERR#. V *south bridge* je umiestnená špeciálna logika, ktorá vyhodnocuje stav týchto signálov. V prípade chyby je transakcia ukončená špecifickým spôsobom. Ten závisí na konkrétnom systémovom softvéri (BIOS) a do určitej miery je programovateľný. Obecne kontrola parity je nedostatočná kontrola, pretože nedokáže detekovať všetky chyby.

3.10 Ukončenie transakcie

Ak cieľové zariadenie nepodporuje burst režim, alebo nie je schopné prijať dáta od iniciátora, musí byť toto zariadenie schopné požiadať o ukončenie transakcie. Využíva sa k tomu signál STOP#. Rozlišujeme tri typy ukončenia transakcie cieľovým zariadením:

Retry (opakovanie): Jedná sa o požiadavku o ukončenie transakcie ešte pred samotnou dátovou fázou, keď cieľové zariadenie nie je schopné dokončiť transakciu. Zariadenie požiada o tento typ ukončenia aktivovaním signálu STOP# a deaktivovaním TRDY# signálu pred zahájením počiatočnej dátovej fázy.

Disconnect (odpojenie): Je špeciálny typ ukončenia transakcie vyvolaný cieľovým zariadením. Cieľové zariadenie môže požiadať o tento typ ukončenie pred samotným začatím alebo počas priebehu dátovej fáze. Nie je zaručené, že cieľové zariadenie akceptovalo dát, preto je potrebné transakciu opakovať. Proces ukončenia je vyvolaný aktivovaním signálov STOP# a TRDY#.

Target-abort (zrušenie): Jedná sa o netypické ukončenie transakcie, keď cieľové zariadenie nie je schopné dokončiť transakciou z dôvodu nejakej chyby. Proces zrušenia transakcie je vyvolaný deaktivovaním signálu DEVSEL# a aktivovaním STOP#.

3.11 Konektory a elektrické vlastnosti PCI

PCI štandard rozlišuje dva rôzne typy zariadení: 3.3V a 5V. Pre tento dôvod existujú dva typy konektorov. Jeden pre 5V napäťovú úroveň a druhý pre 3.3V napäťovú úroveň. Tieto konektory sú kľučované tak aby, nedošlo k použitiu zariadenia na nesprávnu napäťovú úroveň.

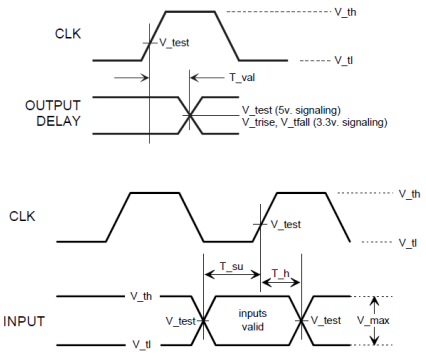
Z dôvodu kompatibility existuje univerzálne zariadenie, ktoré je schopné pracovať s oboma napäťovými úrovňami. Konektor tohto zariadenia musí byť kľučovaný tak, aby mohol používať oba typy konektorov pre 3.3V a 5V a vnútorná logika zariadenia musí byť schopná pracovať s oboma napäťovými úrovňami.

Konštrukcia PCI umožňuje definovať maximálny výkon požadovaný zariadením pomocou signálov PRSNT1 a PRSNT2.

PRSNT1	PRSNT2	Popis
1	1	Slot je prázdny
0	1	Zariadenie prezentuje, 25W max
1	0	Zariadenie prezentuje, 15W max
0	0	Zariadenie prezentuje, 7.5W max

3.12 Časové priebehy

PCI je prísne synchrónna zbernica, kde všetky signály sú viazané na nábežnú hranu hodinového signálu CLK. Zdroj signálu musí generovať príslušný signál na zbernicu s maximálnym oneskorením *Tval*. Prijímač signálu ma zabezpečené, že signál bude na zbernici nastavený najneskôr s oneskorením *Tsu* pred nábežnou hranou CLK (viz. 3.9). *Tval* je 11ns (max) a *Tsu* je 7ns (min). Pri perióde CLK 30 ns zostáva 30-11-7=13ns na dobu šírenia signálu. Zbernica PCI pracuje s odrazenou vlnou a za tento čas musí signál doraziť od zdroja na koniec zbernice, tam sa odrazí a vrátiť späť. Konkrétne hodnoty sú uvedené v dokumentácii PCI zbernice [7].



Obrázek 3.9: Podmienky signálov PCI zbernice.

Kapitola 4

Návrh PCI karty

Kapitola sa zaoberá procesom analýzy a návrhu PCI karty. Rozdelená je na tri hlavné podkapitoly: návrh systému 4.1, návrh hardvéru 4.2 a návrh softvéru 4.4.

4.1 Návrh systému

Cieľom práce je zhotoviť komunikačnú kartu pre počítač PC, ktorá bude slúžiť k ovládaniu servo motorov rezacích strojov. Karta sa zasúva do voľného slotu PCI zbernice a musí podporovať oba typy štandardov PCI: 5V, 3.3V štandard. Komunikácia bude prebiehať pomocou štyroch komunikačných kanálov, kde každý komunikačný kanál bude realizovaný dvoj-prístupovou pamäťou zo strany počítača a jedno-čipovým komunikačným procesorom zo strany servo motorov. Dva kanály budú obsahovať prevodník na rozhranie RS485 - galvanický neoddelený, zvyšne dva kanály budú zakončené sériovým rozhraním UART, ktoré slúži k pripojeniu špeciálnych modulov. Poradie komunikačných kanálov bude možne programovo meniť.

Z pohľadu PCI zbernice by doska mala predstavovať štyri dvoj-prístupové pamäte a niekoľko registrov umiestnených v pamäťovom adresnom priestore. Adresa dvoj-prístupových pamäti a registrov bude absolútna voči bázovej adresy dosky a nie je možne ju meniť. Bazová adresa dosky bude určená obsahom registra BAR (Base adress 0) v konfiguračnom priestore dosky. Reštartom PC systémový softvér počítača prehľadá všetky sloty PCI zbernice a na základe obsahu BAR registra pridelí potrebné systémové prostriedky pre komunikačnú kartu. Hodnota BAR registra bude nastavená tak, aby sa pridelená pamäť nachádzala tesne pod hranicou 32 bitového adresového priestoru (4GB).

Všetky prístupy na komunikačnú dosku zo strany PCI musia byť realizované bajtovými operáciami. Tento fakt je potrebné zohľadniť pri návrhu ovládača dosky.

Veľkosť adresového priestoru dosky bude 4kB a bude rozdelený do nasledujúcej štruktúry:

Adresa 0 až FFh: Na tejto adrese bude namapovaná dvoj-prístupová pamäť pre prvý komunikačný kanál.

Adresa 100 až 1FFh: Na tejto adrese bude namapovaná dvoj-prístupová pamäť pre druhý komunikačný kanál.

Adresa 200 až 2FFh: Na tejto adrese bude namapovaná dvoj-prístupová pamäť pre tretí komunikačný kanál.

Adresa 300 až 3FFh: Na tejto adrese bude namapovaná dvoj-prístupová pamäť pre štvrtý komunikačný kanál.

Adresa 400h: Na tejto adrese bude namapovaný riadiaci register CTRL1. Register slúži k reštartovaniu komunikačných procesov a nadstavenie žiadosti o prerušenie na PCI zbernici.

Adresa 500h: Na tejto adrese bude namapovaný riadiaci register CTRL2. Register slúži k riadeniu prerušenia od komunikačných procesov.

Adresa 600h: Na tejto adrese bude namapovaný stavový register. Register slúži k ovládaniu poradia komunikačných kanálov.

4.2 Návrh hardvéru

Kapitola popisuje návrh komunikačnej karty. Zaoberá sa výberom vhodných komponent a ich integráciou do projektu. V závere kapitoly je uvedená bloková schéma zariadenia, ktorá slúži ako podklad pre implementáciu dosky plošných spojov.

4.2.1 Výber platofrmy

Jadro systému tvorí radič PCI zbernice. Pri jeho hardvérovej realizácii je potrebné brať ohľad na niekoľko faktorov. Dôraz bude kladený najmä na podporu technológie, kvalita vývojového prostredia, dostupnosť na trhu a cena. V teoretickej časti práce je popísaná technológia FPGA obvodov (viz. 2). Táto platforma sa javí najvhodnejšia pre realizáciu PCI radiča.

Výber FPGA obovdou

Výber FPGA obvodov je značne široký. Je potrebné aby vybraný obvod určený pre realizáciu projektu spĺňoval niekoľko zásadných podmienok.

- **Frekvencia** - zbernica PCI pracuje na kmitočte 33 MHz. V budúcnosti je možne počítať s rozšírením komunikačnej rýchlosti na 66 MHz. Vybraný obvod musí tieto frekvencie podporovať.
- **Systémové prostriedky** - jedná sa o počet systémových hradiel a CLB blokov, ktoré realizácia radiča vyžaduje. Hodnota je závislá na zložitosti systému a nie je možné ju presne určiť. Jej výber závisí do istej miery na skúsenosti užívateľa s danou technológiou. Je potrebné počítať s istou rezervou.
- **Veľkosť konfiguračnej pamäte** - parameter úzko súvisí s veľkosťou systémových prostriedkov. Určuje veľkosť konfiguračného reťazca, ktorý je možne nahrať do FPGA obvodu.
- **Veľkosť blokovej pamäte** - využívanie blokovej pamäti šetrí systémové prostriedky FPGA obvodu. Veľkosť musí byť zvolená tak, aby bola dostatočná pre realizovanie registrov a dvoj-prístupových pamätí komunikačného kanálu.
- **Počet vstupno-výstupných pinov** - výber púzdra je potrebné dôkladne zvážiť, aby bol počet vstupno-výstupných pinov dostačujúci.

- **Softvérova podpora** - výrobca by mal mať dostupnú kvalitnú softvérovú podporu a vývojové prostredie.
- **Logické úrovne** - daný obvod by mal byť schopný rozlíšiť logickými úrovne s ktorými pracuje PCI zbernica. Štandard PCI definuje logické úrovne 3.3V a 5V.
- **Cena - Výkon** - výberom obvodu je potrebne vhodne zvážiť pomer cena/výkon, aby nebol použitý zbytočne výkonný obvod. Naopak je potrebné počítať s určitými výkonnostnými rezervami, pre možné rozšírenie funkčnosti karty.

S ohľadom na dané kritéria sa vhodne javí použitie obvodu rodiny Spartan II. Táto rodina obvodov ponúka niekoľko typov zariadení s rôznym počtom vstupno-výstupných pinov. K realizácii dosky bude použitý konkrétny obvod XC2S50 v púzdre PQ208.

Obvod XC2S50

Obvod XC2S50 je dodavaný v dvoch rôznych variantách. Prvá varianta je charakteristická nižšou odolnosťou (teplota, vlhkosť) a je určená pre komerčné zariadenia. Druhá varianta sa využíva v priemyselnom odvetí a je charakteristická vyššou odolnosťou. Pre vyhotovenie karty je potrebné použiť obvod pre priemyselné podmienky.

Základne vlastnosti obvodu XC2S50:

- *Obvod obsahuje 1728 logických buniek.*
- *Obvod obsahuje 50000 systémových hradiel.*
- *Obvod obsahuje 384 CLB blokov.*
- *Obovod obsahuje 140 vstupno-výstupných pinov.*
- *Pracovná teplota: -40 až 100 C.*
- *Obvod obsahuje 8 blokových RAM pamäti o maximálnej veľkosti 32K.*
- *Veľkosť konfiguračného reťazca: 559200 bitov.*
- *Minimálny čas potrebný k zachyteniu zmeny na vstupne: 2,7 ns.*
- *Minimálny čas potrebný k nastaveniu výstupu: 4.4 ns.*
- *Maximálna spotreba: 100 mA. Typicky 12 mA.*

Parametre uvedené v zozname sa vzťahujú pre konkrétny typ obvodu XC2S50. Zvyšné parametre sú pre celú rodinu obvodov Spartan II rovnaké. Uvedené sú v dokumentácii obvodu [18].

Výber mikrokontroléru

Výber mikrokontroléru pre riadenie komunikačných kanálov bude vychádzať z koncepcie dosky určenej pre ISA zbernicu. Využíva sa tam 8-bitový mikroprocesor rady 8051 od firmy Atmel (konkrétne typ AT89C2051 [9]). Tento typ procesoru bude využitý v doske určenej pre PCI zbernicu. Zabráni sa tak vzniku možných chýb, ktoré by mohli vzniknúť vývojom komunikačného softvéru pre nový typ procesoru. Zmena procesoru môže byť podnetom novej revízie dosky.

Mikrokontrolér AT89C2051

Mikrokontrolér AT89C2051 je výkonný 8-bitový procesor s veľkosťou flash pamäte 2kB. Obsahuje 15 vstupno-výstupných pinov, dva 16-bitové čítače a programovateľné sériové rozhranie. Jadro procesoru môže pracovať na frekvencii od 0 - 24 MHz a pracovné napätie môže byť v rozmedzí 2,7 až 6V.

4.2.2 Návrh zdrojovej časti

V návrhu zdrojovej častí musí byť v úvahu brané maximálne výkonnostné zaťaženie PCI zbernice a maximálna spotreba zariadenia. Na základe tých faktov je potrebné rozhodnúť, či zariadenie môže byť napájane priamo zo PCI zbernice, alebo bude potrebné externé napájanie. Je nutné brať v poddotaz, že karta sa bude využíva v priemyselnom odvetí, kde teplota okolia môže byť značne vyššia a môže spôsobiť vyššie zaťaženie zdroja. V návrhu je potrebne zohľadniť, že zariadenie pracuje na vysokých frekvenciách a preto zdroj nesmie byť zdrojom rušivých signálov.

Zdroj PCI zbernice

Systémový konektor PCI zbernice obsahuje štyri rôzne napäťové úrovne: +5V, +3.3V, +12V, -12V. Každá z daných napäťových úrovni má danú maximálnu hranicu zaťaženia viz. tabuľka 4.1. Celkové zaťaženie konektoru zbernice nemôže prekročiť hodnotu 25W.

Úroveň	Zaťaženie
5V	max 5A
3.3V	max 7,6A
12V	max 500 mA
-12V	max 100 mA

Tabulka 4.1: Miera zaťaženia PCI zbernice.

Spotreba zariadnia a napäťové úrovne

Zariadenie bude pracovať s troma rôznymi napäťovými úrovňami:

5V - napäťová úroveň 5V je potrebná pre napájanie mikrokontrolérov a prevodníka RS485. Úroveň sa získa priamo zo systémového konektoru PCI zbernice.

3.3V - napäťová úroveň 3.3V je potrebná pre napájanie FPGA obvodu a konfiguračnej pamäte. Zdrojom tejto úrovne bude systémový konektor PCI zbernice.

2.5V - napäťová úroveň 2,5V je potrebná pre napájanie jadra FPGA obvodu. Úroveň sa získa z 5V úrovne použitím pasívneho regulátora napätia. Regulátor musí byť dimenzovaný tak, aby sa pri plnom zaťažení zariadenia neprekračoval maximálne stanovené hodnoty.

Výkon karty je možne určiť na základe zoznamu použitých obvodov. V dokumentácii každého obvodu je uvedená hodnota maximálneho odberu prúdu v mA (A) pri danej logickej úrovni. Výkon je tak možne vypočítať pomocou vzorca: $P=U*I$. Nasledujúci zoznam uvádza výkonnostné hodnoty jednotlivých obvodov zariadenia:

FPGA - Zdroj pre napájanie FPGA obvodu musí minimálne poskytovať 500 - 700 mA pri napäťovej úrovni 3,3V. To činí výkon 2,3 W. V extrémnych prípadoch môže FPGA obvod krátkodobo zaťažiť hodnotou 6W (1,8A). Je potrebné počítať s týmto faktorom pri voľbe regulátora.

Mikroprocesor - Maximálny odber mikrokontroléru je v dokumentácii stanovený na hodnotu 25.5 mA pri napäťovej úrovni 6V (0,15 W). Maximálny výkon obvodu tak činí 0.6 W.

RS485 prevodník - Odber prevodníka RS485 sa pohybuje rádovo v desiatkach mA. Tento výkon je zanedbateľný pre návrh zdroja zariadenia.

Súčet výkonov obvodov s najvyššou spotrebou dosahuje 7W. Konektor PCI zbernice je možné zaťažiť až do 25 W, preto je možne pre napájanie karty použiť systémový konektor zbernice. Pre napájanie zariadenia sa využijú napäťové úrovne 5V a 3.3V. Napätie 3.3V sa využije pre napájanie FPGA obvodu a konfiguračných pamäti. Napätie 5V sa využije k napájaniu procesorov, prevodníka RS485 a k získaniu napätia úrovne 2.5V.

4.2.3 Konfiguračná pamäť

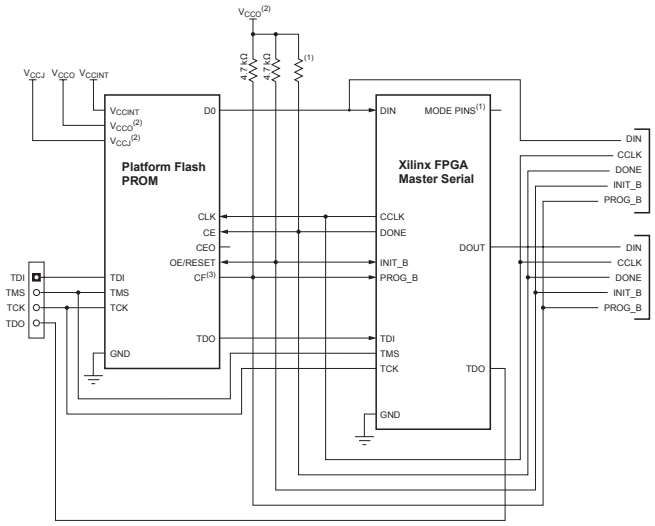
Obsah FPGA obvodov je napäťovo závislý. Po strate napájania je konfigurácia FPGA obvodu vymazaná a pri každom ďalšom spustení je potrebne obvod opäť nakonfigurovať. Pre tento proces je potrebné použiť napäťovo nezávislú externú pamäť, ktorá pri inicializácii (reštarte alebo privedenie napájania) nahrá svoj obsah (konfiguračný reťazec) do pamäte FPGA obvodu. Ako konfiguračná pamäť bude použitá flash pamäť série XC18V00, ktorá umožňuje 20 tisíc konfiguračných cyklov viz. [19]. Pre FPGA obvod XC2S50 bude použitý konkrétny typ XC18V01 s veľkosťou pamäte 1 MB. Pamäť bude zapojená v sérii v režime master, kde konfiguračný proces bude riadený FPGA obvodom. Nadstavenie FPGA obvodu v konfiguračnom režime master, sériové zapojenie je potrebné správne nasstaviť *pull-up* rezistory na pinoch M0 - M2 FPGA obvodu. Schéma zapojenia je uvedená na obrázku 4.1. Zapojenie vychádza zo štandardného zapojenia uvedeného v dokumentácii pamäte.

Obvody sú zapojene v reťazci, kde na vstupe je umiestnená konfiguračná pamäť a na výstupe FPGA obvod. Programovanie bude prebiehať pomocou rozhrania JTAG. Je tak možne programovať zvlášť konfiguračnú pamäť a zvlášť FPGA obvod.

Pri inicializácii FPGA obvodu (privedením napätia alebo aktivovaním vstupu PROGRAM na logickú úroveň 0) obvod začne generovať hodinový signál CCLK a nastaví potrebné signály pre inicializáciu konfiguračnej pamäte. Obsah konfiguračnej pamäte je na základe hodín CCLK postupne vysielaný na výstupný pin DO. Proces prenos dát je kontrolovaný pomocou CRC. V prípade chyby FPGA obvod generuje prerušenie a prenos reťazca je ukončený. Prenosová frekvencia bude 33 MHz. Počas konfiguračného cyklu je signál DONE v logickej úrovni 0. Po úspešnom ukončení prenosu je tento signál nastavený na hodnotu 1.

4.2.4 Rozhranie RS485 a UART

Rozhranie RS485 [10] je prenosový komunikačný štandard definovaný v roku 1983 inštitúciou EIA. Je tvorený dvoj-vodičovým vedením označených A, B (niekedy +,-). Signál je šírený pomocou týchto vodičov tak, že na jednom vodiči je úroveň H a na druhom úroveň L. Logická úroveň sa tak určí na základe rozdielu napätí medzi vodičmi. Hodnota H je reprezentovaná minimálnym rozdielovým napätím A - B $>$ 300 mV (kladný rozdiel) a hodnota L je reprezentovaná záporným rozdielovým napätím A - B $>$ - 300 mV. Správny vysielač



Obrázek 4.1: Schéma zapojenia konfiguračnej pamäti.

by mal na výstupe generovať napätie s minimálnym rozdielom 2 V a správny prijímač by mal na vstupe rozlíšiť napätie s rozdielom 200 mV. Tento typ prenosu je značne odolný voči rušeniu a umožňuje viesť signál na dĺžku 1200m pri maximálnej prenosovej rýchlosti 10 MB/s.

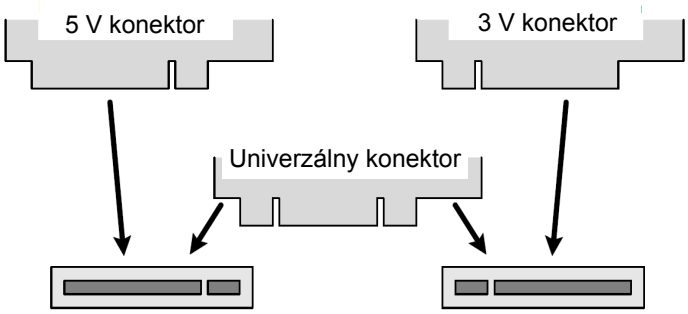
Sériové rozhranie UART [8] je prenosový komunikačný kanál určený najmä pre mikrokontroléry. Jedná sa o asynchrónny typ komunikácie, kde posielane dáta nie sú riadene žiadnym hodinovým signálom. Komunikácia je synchronizovaná špeciálnymi bitmi, ktoré sú vkladané do každého vysielaného slova. Vysielané slovo má presne stanovenú dĺžku a môže byť doplnené ďalšími informačnými bitmi. Vysielač a prijímač musia mať synchronizované vlastným hodinovým signálom s rovnakou frekvenciou tak, aby bolo možne rozlišovať dĺžku jednotlivých logických úrovni. Rozhranie UART využíva 5V TTL logickú úroveň.

Komunikačná karta bude obsahovať 4 komunikačné kanály. Dva kanály budú tvorené sériovým rozhraním UART a budú slúžiť pre pripojenie externých modulov. Ďalšie dva kanály budú obsahovať prevodník na rozhranie RS485. Rozhranie bude zakončené konektorom CANNON9. Rozhrania RS485 bude galvanicky neoddelene a je potrebné implementovať na výstupe prepäťovu ochranu aby prípadná napäťová špička nezničila obvody zariadenia.

4.2.5 Konektor PCI zbernice

Konektor PCI zbernice musí byť navrhnutý tak, aby umožňoval prenášať informácie o frekvencii 33 MHz a bol značne mechanický odolný. Štandard PCI zbernice definuje presne mechanické rozmery konektoru z dôvodu kompatibility zariadení. Rozlišujú sa dva základne typy konektorov: female (zásuvka) a male (zástrčka). Female je konektor výlučne spojený so systémovou doskou počítača. Male je typ konektoru výlučne spojený s cieľovými zariadeniami. Niektoré zariadenia však môžu súčasne obsahovať konektor male a female. Jedna sa väčšinou o zariadenia slúžiace k analýze zbernice.

PCI štandard ďalej definuje konektory pre 3.3V logiku, 5V logiku a univerzálny typ konektoru pre oba typy logických úrovni. Tieto typy sa líšia v pozícií kľúča v konektore. Univerzálny konektor zahrnuje pozíciu oboch kľúčov viz. obrázok 4.2.

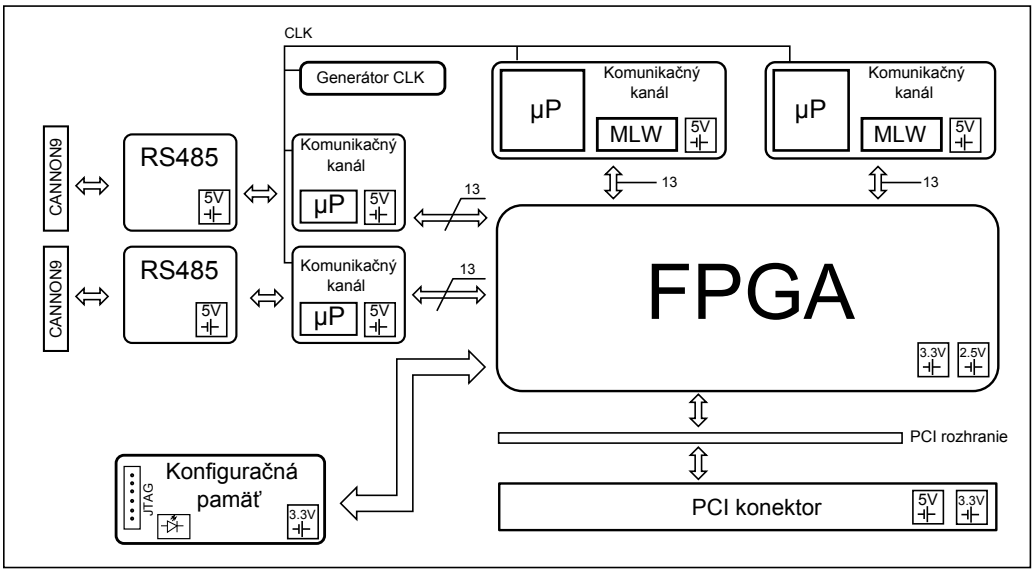


Obrázek 4.2: Typy konektorov PCI zbernice [18].

Zariadenie musí byť schopné pracovať s oboma typmi napäťových štandardov, preto bude použitý univerzálny typ PCI konektoru.

4.2.6 Bloková schéma

Jednotlivé časti popísane v predchadzajucích kapitolách je možné pospájať do jedného funkčného celku. Výsledkom je bloková schéma uvedená na obrázku 4.3.

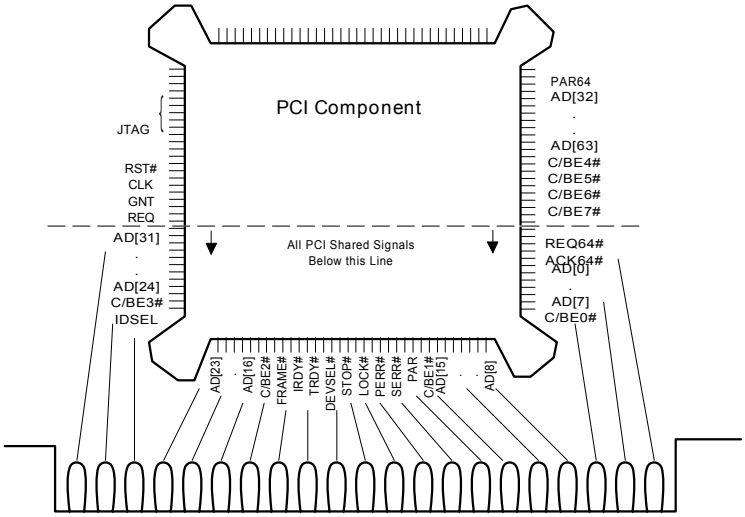


Obrázek 4.3: Bloková schéma obvodu.

Jadrom systému je blok *FPGA*, ktorý je tvorený obvodom Spartan II. Tento obvod riadi komunikáciu na PCI zbernici. Prepojenie medzi blokmi *FPGA* a *PCI konektorom* musí spĺňovať niekoľko základných kritérií:

- Dĺžka vodiča signálu hodín CLK nesmie prekročiť limit maximálnej dĺžky 2.5 palca.

- Vodiče signálov pre riadenie zbernice a dátové signály nesmú presiahnuť dĺžku 1.5 palca.
- Štandard PCI doporučuje fyzické prepojenie signálov medzi konektorom PCI a obvodom FPGA. Postup zapojenia je vhodné dodržať viz. obrázok 4.4.



Obrázek 4.4: Prepojenie PCI zbernice s FPGA obvodom [18].

Výstupom bloku *FPGA* sú štyri komunikačné kanály, ktoré sú tvorené mikroprocesormi Atmel. Každý komunikačný kanál je prepojený s FPGA obvodom signálmi, ktorými mikroprocesor pristupuje do dvoj-portovej pamäte FPGA obvodu. Výstupom dvoch komunikačných kanálov bude sériové asynchrónne rozhranie UART, ktoré bude zakončené konektorom MLW10G. Smer komunikácie bude určovať riadiaci signál od mikroprocesoru. Zvyšne dva komunikačne kanály sú prepojené s výstupným blokom *RS485*. Komunikácia medzi týmito blokmi je tvorená sériovým asynchrónnym rozhraním UART a jedným riadiacim signálom. Zdrojom hodinového signálu pre komunikačné kanály bude použitý kryštál o frekvencii 22 MHz. Tento hodinový signál bude postupne distribuovaný k všetkým komunikačným kanálom.

Blok *konfiguračná pamäť* musí byť umiestnený v blízkosti FPGA obvodu tak, aby nedochádzalo k prenosovým chybám. Na vstupe bloku je šesť pinový konektor, ktorý slúži k pripojeniu JTAG programátora. Proces konfigurácie bude signalizovaný červenou LED diódou.

RS485 je výstupný blok zakončený konektorom CANNON9. Je tvorený obvodom LTC485 [3], ktorý realizuje prevodník UART - RS485. Na výstupe rozhrania budú dva rezistory, ktoré slúžia ako pojistky a zenerovové diódy, ktoré slúžia ako prepäťová ochrana. Rozhranie RS485 bude ukončene prepínačmi pre zakončenie komunikačnej linky.

Jednotlivé bloky pre svoju správnu funkčnosť potrebuje napájacie napätie. V blokovej schéme uvedenej na obrázku sú uvedené napäťové úrovne pre jednotlivé bloky. Súčasťou zdrojovej časti sú filtrovacie a blokovacie kondenzátory. Ako filtrovacie kondenzátory budú použité keramické kondenzátory o veľkosti 100nF. Tieto kondenzátory slúžia na potlačenie rušivých signálov a budú umiestnené v blízkosti napájania obvodov. Ako blokovacie

kondenzátory budú použite tantalové kondenzátory o veľkosti 10uF. Tieto kondenzátory budú rovnomerne rozložené po doske. Zabráni sa tak poklesu napätia pri vyššom zaťaženie zdroja. Pre získanie napätia 2.5V bude použitý lineárny regulátor napätia REG1117A [4].

4.3 Analýza komunikačného protokolu

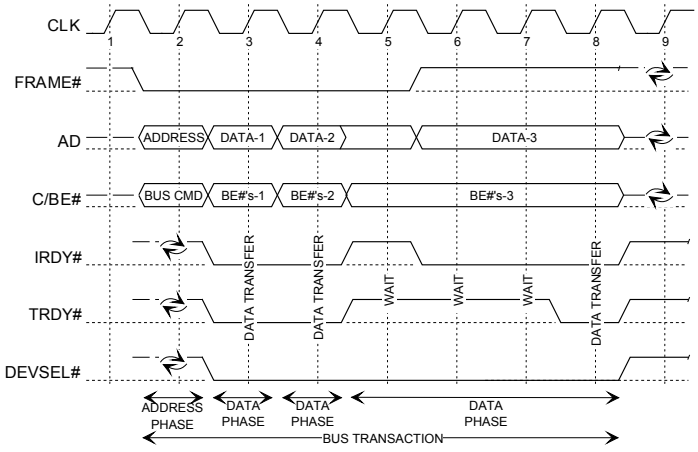
Komunikácia na PCI zbernici prebieha pomocou transakcii, kde transakcia predstavuje sekvenciu vykonaných krokov počas ktorých sú prenesené dáta po zbernici. PCI štandard definuje niekoľko typov transakcii a nie všetky transakcie musí zariadenie nevyhnutne podporovať. V nasledujúcom výklade budú popísané transakcie, ktoré sú potrebné pre minimálny chod komunikačnej karty.

4.3.1 Pamäťové prenosy

Pamäťový prenos je prenos dát medzi iniciátorom a cieľovým zariadením, kde iniciátor pristupuje do pamäte zariadenia. U komunikačnej karty pamäť zariadenia bude predstavovať štyri dvoj-portvé pamäte, stavové a riadiace registre. PCI štandard definuje dva rôzne typy prenosu dát: čítanie a zápis dát.

Pamäťový zápis

Na obrázku 4.5 je znázornený priebeh signálov pamäťového zápisu. Jedná sa o blokový prenos dát, kde iniciátor zapisuje dáta do cieľového zariadenia. Predpokladá sa, že zariadenie už má pridelenú bázovú adresu.



Obrázek 4.5: Pamäťový zápis [18].

Na počiatku transakcie (druhá hrana hodinového signálu) iniciátor aktivuje signál FRAME#, čím urči, že na zbernici je platná adresa a riadiace bity C/BE#. Ak cieľové zariadenie rozpozná svoju adresu na zbernici a identifikuje príkaz (bity C/BE#) aktivuje signál DEVSEL#. Podľa toho, na ktorý hodinový cyklus cieľové zariadenie aktivuje signál DEVSEL# rozlišujeme zariadenie typu: *fast, medium a slow*. Na obrázku je uvedené zariadenie typu *fast*, signál je aktívny na druhý takt po dekódovani adresy.

Samotný priebeh prenosu dát je riadený signálmi TRDY# a IRDY#. Signál TRDY# je riadený cieľovým zariadením a jeho aktivovaním zariadenie signalizuje, že je pripravené na prenos dát. Naopak, deaktivovaním signálu, je prenos dát pozastavený po dobu maximálne osem hodinových cyklov alebo pokým signál nie je opäť aktivovaný. Signál IRDY# je riadený iniciátorom a jeho význam je obdobný ako signál TRDY#.

Prvá dátová fáza začína na tretiu hranu hodinového signálu. Na obrázku je vidieť aktívne signály TRDY# a IRDY#. Prenos dát trvá dva hodinové takty (3. a 4. takt) a budú tak prenesené dva dátové bloky. Na piatu hranu hodinového signálu sú deaktivované signály TRDY# a IRDY#. Iniciátor a cieľové zariadenie tak signalizujú, že nie sú pripravené na prenos dát a transakcia tak prechádza do čakacieho stavu. Signál IRDY# je aktivovaný na šiestu hranu hodinového signálu. Znamená to, že iniciátor je pripravený k prenosu dát a vystavuje nové dáta na zbernici. Cieľové zariadenie však nie je ešte pripravené akceptovať dáta. Iniciátor preto musí platne dáta na zbernici držať dokým zariadenie nie je pripravené k prenosu. Na ôsmu hranu hodinového signálu cieľové zariadenie aktivuje signál TRDY# a vyčíta dáta na zbernici.

Cieľové zariadenie nikdy, nevie koľko dát sa bude prenášať a preto je potrebné signalizovať koniec prenosu. Posledná dátová fáza je určená deaktivovaním signál FRAME# a aktivovaním signálu IRDY#. Na obrázku 4.5 posledná dátová fáza nastane na ôsmu hranu hodinového signálu. Signály IRDY#, TRDY# a DEVSEL# musia byť budené a deaktivované ešte jeden hodinový takt po ukončení dátovej fáze.

Počas celej doby prenosu signály C/BE# indikujú platnosť dát v dátovom dvoj-slove. Signály sú riadene iniciátorom a na zbernici sú vystavené vždy takt potom, ako cieľové zariadenie akceptuje dáta z predošlej dátovej fázy. To znamená, že signál C/BE# sú nastavené na zbernici aj v prípade, že zariadenie sa nachádza v čakacom stave. Výnimku tvorí prvá dátová fáza, kde iniciátor nastaví signály spolu s dátami viz. 4.5.

Počas jednoduchého dátového prenosu (jedna dátová fáza) je princíp komunikácie rovnaký ako u blokového prenosu. Rozdiel je v tom, že iniciátor aktivuje signál FRAME# iba po dobu jedného hodinového taktu. Cieľové zariadenie tak po dokončení prvej dátovej fáze detekuje neaktívny signál FRAME# a ukonči prenos. Jednoduchý dátový prenos bude popísaný v kapitole 4.3.2.

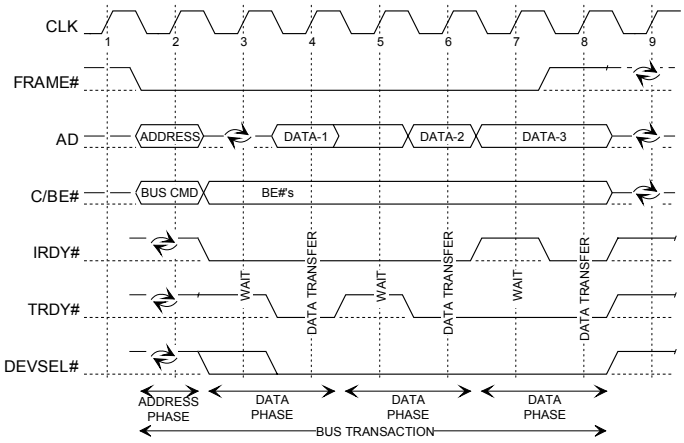
Pamäťové čítanie

Na obrázku 4.6 je znázornený priebeh transakcie pamäťového čítania. Jedná sa o blokový prenos dát a predpokladá sa, že zariadenie už má pridelenú bázovú adresu. Režim čítanie dát sa len mierne odlišuje od zápisu dát.

Transakcia začína aktivovaním signálu FRAME# na druhu hranu hodinového signálu. Signál FRAME# určí, že na zbernici sa nachádza platná adresa a signály C/BE#. Ak cieľové zariadenie rozpozná svoju adresu na zbernici a identifikuje príkaz C/BE# nadstaví signál DEVSEL#.

Na tretiu hranu hodinového signálu iniciátor aktivuje signál IRDY# a signalizuje tak, že je pripravený k prenosu dát. Aby cieľové zariadenie mohlo zapísať dáta na zbernicu, je potrebné aby iniciátor uvoľnil zbernicu. Deje sa tak vložením tzv. vratného cyklu na tretiu hranu hodinového signálu. Iniciátor nadstavuje príslušne signály (AD[31:0]) do stavu vysokej impedancie.

Na štvrtú hranu hodinového signálu cieľové zariadenie aktivuje signál TRDY# a určí tak, že na zbernicu sú vystavené platné dáta. Iniciátor môže prečítať dáta zo zbernice. Na piatu hranu hodinového signálu je deaktivovaný signál TRDY# a cieľové zariadenie tak



Obrázek 4.6: Pamäťové čítanie [18].

urči, že na zbernicu nestihol vystaviť platné dáta. Iniciátor tak musí čakať, dokým nie je opäť aktívny signál TRDY# a na zbernici nie sú platne dáta. Deje sa tak na šiesty hodinový cyklus, kde iniciátor detekuje aktívny signál TRDY# a číta dáta zo zbernice.

Signály C/BE# sú významovo rovnaké ako u zápisovej transakcie. Signály sú riadene iniciátorom a platne dáta obsahujú vždy jeden hodinový takt v predstihu pred dátovou fázou. Na uvedenom obrázku tento fakt nie je možne vidieť, pretože sa jedná o blokový prenos, kde typicky bývajú aktívne všetky bajty.

Prenos dát je ukončený deaktivovaním signálu FRAME# a aktivovaním signálov TRDY# a IRDY# na ôsmu hranu hodinového signálu. Signály IRDY#, TRDY# a DEVSEL# musia byť budené a deaktivované ešte jeden hodinový takt po ukončení dátovej fáze.

4.3.2 Konfiguračný prenos

Konfiguračný prenos je špecialny prenos dát medzi inicátorom a cieľvoým zariadenim, kde iniciátor pristupuje do konfiguračnej pamäte zariadenia. Najčastejšie sa jedná o jednoduchý typ prenosu dát (jedna dátova fáza).

Štruktára konfiguračnej pamäte je pevne daná PCI štandardom a jej veľkosť je 256 bajtov. Pre základnú funkčnosť komunikačnej karty budú použité iba povinné registry (viz 3.5) a register bázovej adresy. Ostatné registry sú voliteľné a nebudú sa využívať.

PCI štandard definuje niekoľko typov konfiguračných prenosov. Pre implementaciu cieľového zariadenia je potrebný typ nula (viz. 4.3.2). Ostatné typy sa netýkajú cieľových zariadeni a ich implementácia bude zanedbaná.

Prenos typu nula

Podobne ako u pamäťových prístupov, tak aj u konfiguračného prenosu typu nula rozlišujeme dva typy transakcii: konfiguračné čítanie a konfiguračný zápis. Na obrázku 4.7 je uvedený priebeh konfiguračného čítania, ktorý sa od pamäťového prenosu odlišuje v adresnej fáze. V procese konfiguračného prenosu sa pre určenie cieľového zariadenia nepoužíva porovnanie na základe bázovej adresa, ale signálu IDSEL. Ak cieľové zariadenie detekuje

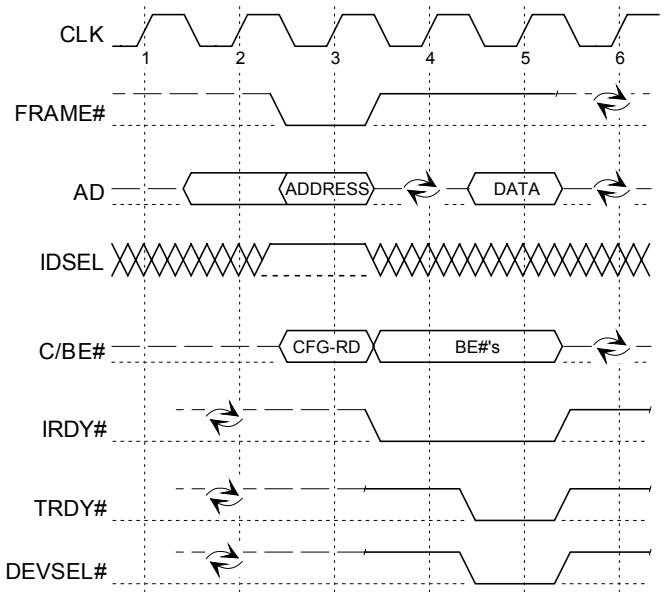
súčasne aktívny signál IDSEL a FRAME#, tak vie, že transakcia bude patriť jemu a aktivuje signál DEVSEL#. Bity C/BE# určujú či sa jedná o čítanie alebo zápis. Bity AD[31:0] budú mať špeciálny formát:

AD[1:0] - definujú typ konfiguračného prenosu. Ak sú bity *00* jedná sa o prenos typu nula.

AD[7:2] - určujú adresu v konfiguračnom priestore.

AD[10:8] - definujú funkciu cieľového zariadenia.

AD[31:11] - bity výhradne využité PCI mostom. PCI radič na základe bitov AD[15:11] vyberá konkrétne fyzické púzdro, ktoré je cieľom konfiguračného prenosu (aktivuje signál IDSEL).



Obrázek 4.7: Konfiguračne čítanie [18].

Dátová fáza je rovnaká ako u pamäťového zápisu. Signály C/BE# určujú platnosť dát v dátovom dvoj slove.

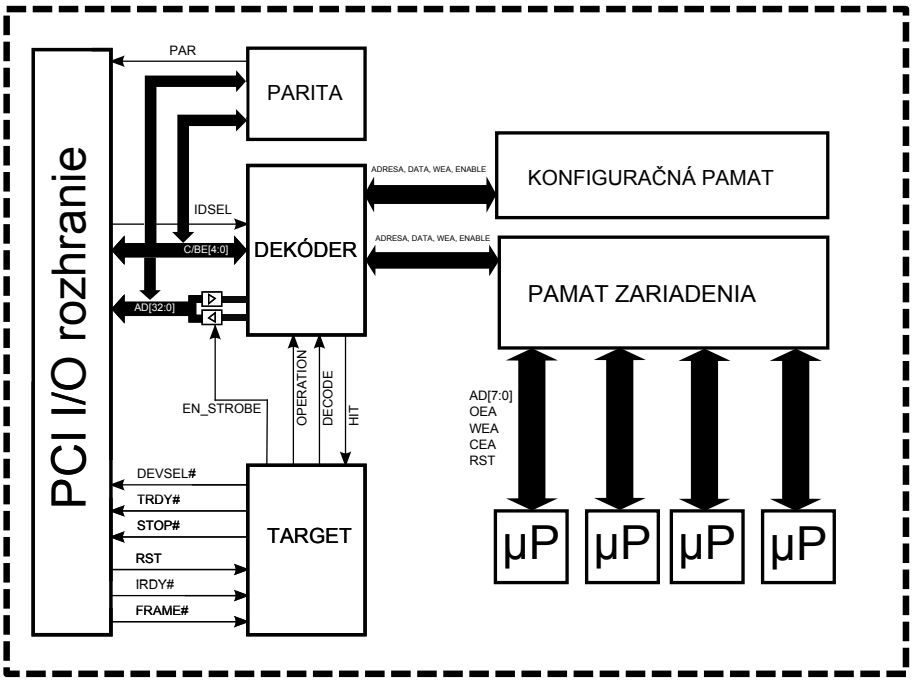
4.4 Návrh radiča

Návrh radiča zbernica vychádza z dokumentácie Xilinx [17], internetového zdroja [6] a štandardu PCI zbernice [18].

Komunikačná karta bude v hierarchii PCI tvoriť vždy cieľové zariadenie (target) a nie je tak potrebné implementovať blok pre riadenie komunikácie z pohľadu iniciátora.

4.4.1 Bloková schéma

Na základe analýzy komunikačného protokolu bola zostrojená nasledujúca bloková schéma viz 4.8.



Obrázek 4.8: Bloková schéma radiča zbernice.

Schéma je tvorená niekoľkými funkčnými blokmi, kde jednotlivé bloky komunikujú pomocou presne definovaných pravidiel. Bloky *decoder, target a parita* sú spojené s PCI zbernicou a riadia komunikáciu na nej. Bloky *konfigurčný priestor a pamäť zariadenia* tvoria hlavnú pamäť zariadenia a ich činnosť je riadena *dekodérom* zariadenia.

Jadro systému tvorí blok *target*, ktorý je realizovaný pomocou stavového automatu. Automat vyhodnocuje riadiace signály PCI zbernice a ovláda závislé bloky v systéme. Podrobný popis je uvedený v kapitole 4.4.2.

Blok *decoder* je prepojený so zbernicou pomocou signálov AD a C/BE#. Tieto signály vyhodnocuje a na ich základe riadi činnosť konfiguračnej pamäte a pamäte obvodu. Popis bloku je uvedený v kapitole 4.4.3.

Blok *parity* slúži k výpočtu parity na rozhraní AD a C/BE#. Činnosť výpočtu parity prebieha iba na konci dátovej fázy a musí byť riadená blokom *target*.

Konfiguračný priestor a pamäť zariadenia tvoria vnútornú pamäť komunikačnej karty. Prístup do jednotlivých typov pamätí je riadený pomocou *dekóderu*.

4.4.2 Target

Target je tvorený stavovým automatom, ktorého návrh vychádza zo špecifikácie PCI. Navrhovaný automat bude mealyho automat [1] a jeho činnosť bude závislá na nábežnú hranu hodinového signálu CLK. Pred samotným návrhom automatu je potrebné definovať prechodové funkcie (vstupno-výstupné podmienky) a stavy automatu.

[1] Typ automatu, kde výstup je generovaný na základe vstupu a súčasného stavu automatu.

Vstupné podmienky

Vstupné podmienky predstavujú vstupné parametre prechodových funkcii automatu. Zo strany PCI zbernice to budú signály riadene iniciátorom: FRAME#, IRDY# a RESET a zo strany vnútornej logiky to bude signál HIT riadený *dekóderom*. Signál HIT bude určovať, či transakcia na zbernici patrí danému zariadeniu viz. 4.4.3.

Výstupné podmienky

Výstupné podmienky tvorí množina signálov, ktoré sú výstupom prechodových funkcii automatu. Riadia priebeh transakcii na zbernici a vnútornú logiku zariadenia. Z pohľadu PCI zbernice sa jedná o signál DEVSEL#, STOP# a TRDY#. Z pohľadu vnútornej logiky sú to signály pre riadenie *dekodéru a parity*.

Vnútorné signály:

- **DECODE** - signál určuje, že *dekóder* môže dekódovať adresu vystavenú na zbernici. Deje sa tak vždy na počiatku transakcie. Počas priebehu transakcie musí byť tento signál deaktivovaný.
- **OPERATION** - signál určuje, že na zbernici prebieha dátová fáza.
- **ENABLE_PARITY** - určí, že v danom okamžiku je potrebné nadstaviť signál PAR na rozhraní PCI zbernice.
- **EN_STROBE** - Zbernica PCI je zdieľaná a je potrebné riadiť budenie signálov na zbernici. Ak daná transakcia nepatrí cieľovému zariadeniu, výstupne signály musia byť zo zbernice odpojené. Signál ENSTROBE bude použitý k riadeniu vstupno-výstupných budičov.
- **EN_STROBE_ADDR** - signál riadi vloženie vratného cyklu na zbernicu. Jeho aktivovaním sú vstupné bloky zmenné na výstupne. Signál je aktívny iba jeden hodinový tak počas transakcie čítania.

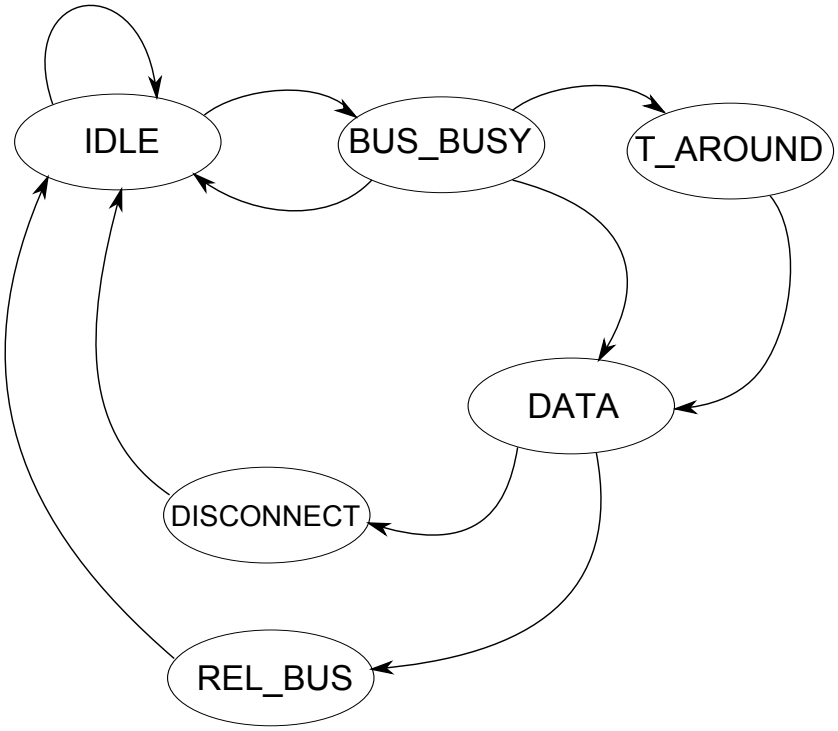
Stavy automatu

Automat je tvorený šesticou stavov:

1. **IDLE**: Počiatočný stav automatu. Každá začatá transakcia musí začínať v stave IDLE. Automat prejde do stavu IDLE vždy po systémovom reštarte alebo po dokončení transakcie.
2. **BUS_BUSY**: Stav automatu, keď zariadenia na zbernici detekuju začatú transakciu (zariadenie detekuje aktívny signál FRAME#). V tomto stave ešte zariadenie nevie komu bude transakcia patriť.
3. **T_AROUND**: Špeciálny stav automatu, keď je potrebne na zbernici vykonať tzv. vratný cyklus.
4. **DATA**: Stav automatu, keď na zbernici sú vystavené platné dáta.
5. **DISCONNECTED**: Koncový stavu automatu identifikujúci ukončenie prenosu. V ďalšom takte automat prechádza do počiatočného stavu.

6. **REL_BUS**: Neočakavaný koncový stavu automatu. Môže nastať v prípade nekonzistentných dát na zbernici. Prenos je ukončený a automat prechádza do počiatočného stavu IDLE.

Na základe analýzy priebehov komunikačného protokolu je možné definovať prechodové funkcie automatu. Výsledkom bude automat uvedený na obrázaku 4.9.



Obrázek 4.9: Stavový automat.

Prechodové funkcie:

IDLE:

goto IDLE	if FRAME# = 1
goto BUS_BUSY	if FRAME# = 0

BUS_BUSY:

goto IDLE	if HIT = 1
goto DATA	if HIT = 1 and write = 1
goto T_AROUND	if HIT = 1 and read = 1

DATA:

goto DISCONNECT	if IRDY# = 0 and FRAME# = 1
goto REL_BUS	if next_state = DISCONNECT

T_AROUND:

goto DATA

DISCONNECT:
goto IDLE

REL_BUS:
goto IDLE

4.4.3 Dekóder

Blok *dekóder* slúži k spracovaniu dát na zbernici a prepína ich na do vnútornej logiky zariadenia. Na vstup bloku sú privedené signály PCI zbernice AD a C/BE# a IDSEL.

Ak sa jedná o pamäťový prístup, blok musí počas adresnej fázy transakcie dekódovať vystavenú adresu na zbernici a porovnať ju s bázovou adresou zariadenia. V prípade zhody aktivuje signál HIT a uloží hodnotu signálov AD do vnútorného registra. Táto hodnota bude využitá k adresovaniu vnútornej pamäte. V ďalšom takte nastane na zbernici dátová fáza. *Dekodér* vyčíta dáta zo zbernice a na základe signálov C/BE# (Byte enable) určí platné dáta. Tieto dáta nastaví na výstupne rozhranie (rozhranie s pamäťou zariadenia). Súčasne nadstaví na výstupe aj adresu uloženú v registre a signály ENABLE a WEA, ktoré určia, či iniciátro bude z vnútornej pamäte čítať alebo do nej zapisovať.

Ak sa jedná o konfiguračný cyklus, dekóder musí rozhodnúť či konfiguračný cyklus patrí cieľovému zariadeniu a nadstaviť tak príslušné signály na výstupnom rozhraní. V tomto prípade sa bude jednať o rozhranie s konfiguračnou pamäťou.

4.4.4 Blok parity

Výstupom bloku je signál PAR. Tento signál riadi paritu na signáloch AD a C/BE. Paritu je potrebné nadstaviť vždy po dokončení transakcii čítania. Pre výpočet je použitá funkcia XOR.

4.4.5 Konfiguračná pamäť

Konfiguračná pamäť musí byť tvorená štruktúrou, ktorá je popísaná v kapitole 3.5.1. Je potrebné aby obsah konfiguračnej pamäte po systémovom reštarte bol nadstavaný na pôvodné hodnoty. Nie všetky registre pamäte je možne prepisovať. Tento fakt je potrebne zohľadniť pri návrhu dekóderu. V prípade bunky bázovej adresy je umožnnený zápis iba do určitých bitov registra. O ktoré bity sa jedná určí množstvo pamäte, ktoré je potrebné alokovať.

Pamäť je pripojená s blokom *dekóder* pomocou signálov ADRESA, DATA, WEA a ENABLE.

4.4.6 Pamäť zariadenia

Pamäť zariadenia bude realizovaná podľa návrhu uvedeného v kapitole 4.1. Pre realizáciu bude použitá dvoj-portová pamäť, aby bol možné vzájomne pristupovať k dátam zo strany *dekóderu* a zo strany komunikačných kanálov blok *uP*. Prístupové rozhranie zo strany *dekóderu* bude realizované pomocou signálov ADRESA, DATA, WEA a ENABLE. Šírka adresnej zbernice musí byť dostatočná, aby bolo možne adresovať priestor o veľkosti 4kB. Šírka dátovej zbernice bude 8 bitov z dôvodu použitia 8-bitového mikroprocesoru. Blok pamäte bude rozdelný na niekoľko častí po veľkosti 512kB. Prvé štyri časti budú realizovať

pamäť pre komunikačné kanály a zvyšné časti budú tvoriť riadiace registre. Nie celá pamäť je využitá.

Zo strany komunikačných kanálov (blok uP) bude prístup do pamäte realizovaný pomocou signálov AD, WE, OE a CE. Adresa a dáta sú multiplexované na spoločnú zbernicu AD. Signály WE, OE a CE sú využité pre riadenie prenosu. Priebeh signálov pre pamäťové čítanie a zápis je uvedený v dokumentacii pamäte DS1609 (viz. [2]). Táto pamäť bola použitá v predošlom dizajne.

4.4.7 Popis riadiacích registrov

1. **Adresa 400h - CTRL1** - Riadiaci register 1 , slúži k reštartovaniui komunikačných procesov. Register je R/W, s vínimkou bitu D7, ktorý je určený len pre čítanie.

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
GIRQ	INT2	INT1	INT0	RST3	RST2	RST1	RST0

Bit **GIRQ** je obraz žiadosti dosky o prerušenie na zbernicu PCI:
GIRQ = L doska MCC4B negeneruje žiadosť o prerušenie.
GIRQ = H doska MCC4B generuje žiadosť o prerušenie.

Bity **INT2** až **INT0** slúžili na predchádzajúcej doske MCC4 (pre zbernicu ISA) a určovali použité prerušenie. Zbernica PCI používa iný mechanizmus priradenia prerušení tak tieto bity sú nevyužité (pri čítaní vracajú úroveň L).

Bity **RST3** až **RST0** slúžia k reštartovaniu komunikačných procesov.
RSTx = L komunikačný procesor je reštartovaný.
RSTx = H komunikačný procesor nie je reštartovaný. Normálna činnosť.

2. **Adresa 500h - CTRL2** - Riadiaci register 2 , slúži k riadeniu prerušení od komunikačných procesov a ovládanie LED D7 až D9.

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
LED D9	LED D8	LED D7	GEN	EN3	EN2	EN1	EN0

Bit **LED D9** až **LED D7** riadia indikačné ledky umiestnené na okraji dosky.
LEDx = L dióda nesvieti.
LEDx = H dióda svieti.

Bit **GEN** riadi globálne povolenie prerušenia z dosky:
GEN = L doska nemôže generovať prerušenie.
GEN = H doska môže generovať prerušenie.

Bit **EN3** až **EN0** povoľujú pre jednotlivé komunikačné procesy žiadosť o prerušenie.
ENx = L komunikačný proces nemôže generovať žiadosť o prerušenie.
Enx = H komunikačný proces môže generovať žiadosť o prerušenie.

Po nábehu napájacieho napätia a po systémovom reštarte je stav registra 00h - LED nesvietia a prerušenie zablokované.

3. **Adresa 600h - STATUS** - Stavový register, slúži k indikácii prerušenia od komunikačných procesov a poradie kanálov.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	COM_ORD	IRQ3	IRQ2	IRQ1	IRQ0

Bit **COM_ORD** riadi poradie komunikačných kanálov v doske (priradenie kanálu k proceosru):
COM_ORD = L štadndardné poradie komunikačných kanálov.
COM_ORD = H obratené poradie komunikačných kanálov.

Bit **IRQ3** až **IRQ0** indikuju stav žiadosti o prerušenie pre komunikačné kanály:
IRQx = L komunikačný procoes X nežiada o prerušenie.
IRQx = H komunikačný procoes X žiada o prerušenie.

4. **Adresa 700h - VERSION** - Stavový verzie, udáva verziu obsahu Xilinx v doske. Určený iba pre čítanie.

D7	D6	D5	D4	D3	D2	D1	D0
VER7	VER6	VER5	VER4	VER3	VER2	VER1	VER0

4.5 Zhodnotenie návrhu

Proces návrhu naznačuje spôsob realizácie komunikačnej karty a predstavuje nevyhnutný krok pred samotnou implementáciu zariadenia. Rozdelený je do dvoch hlavných časti, kde prvá časť popisuje návrh hardvéru a druhá časť popisuje návrh softvéru. Tieto dve časti spolu úzko súvisia a ich implementácia je nevyhnutná pre funkčnosť zariadenia.

Kapitola 5

Návrh dosky plošných spojov

Kapitola popisuje fyzickú realizáciu komunikačnej karty PCI. Jej cieľom je popísať návrh schématu a dosky plošných spojov. Navrhnutý plošný spoj a schémata sú dostupne v prílohe tejto práce vo formate návrhového systému EAGLE.

5.1 Použité technológie

5.1.1 Návrhový systém EAGLE

EAGLE (Easily Applicable Graphical Layout Editor) je návrhový systém nemeckej firmy CadSoft [1]. Jedná sa o výkonný nástroj určený pre realizáciu plošných spojov (PCB). Program obsahuje tri hlavné moduly:

1. **Editor schém** - umožňuje realizovať obvodovú schému zapojenia. K realizácii obvodovej schémy je možne použiť integrované knižnice, ktoré obsahujú zoznam základných súčiastok. Pomocou editora knižníc je možné vytvárať vlastné knižnice a tak značne rozšíriť možnosti návrhu. Schému zapojenia je možné rozdeliť na 999 listov. Program umožňuje základnú kontrolu pravidiel zapojeného obvodu.

2. **Editor spojov** - umožňuje konštruovať až 16 vrstvový plošný spoj s maximálnym rozlíšením 1/10000 (mm). Súčasťou editora sú nástroje na kontrolu návrhu plošného spoja.

3. **Autorouter** - je proces, ktorý umožňuje automatizovaný návrh dosky plošných spojov s využitím stratégie nastaviteľnej užívateľom.

5.2 Popis zapojenia

Návrh obvodového zapojenia vychádza z návrhu popísaného v kapitole 4.2.

Na obrázku v prílohe A je uvedené obvodové zapojenie komunikačnej karty PCI. Jadro systému tvorí FPGA obvod XC2S50 (U1), ktorý je prepojený s PCI zbernicou pomocou definovaného rozhrania. Signály tohto rozhrania sú zapojené v doporučenom poradí podľa obrázku 4.4.

K FPGA obvodu sú ďalej pripojené štyri mikroprocesory, ktoré realizujú komunikačné kanály. Každý komunikačný kanál je s obvodom prepojený pomocou riadiacich a dátových signálov. Signál RST slúži k reštartovania komunikačných kanálov. Signál INT slúži k žiadosti o prerušenie od procesoru na zbernicu PCI a jeho funkčnosť nebude využitá. Signály

WEA, CEA a OEA riadia prístup do pamäte FPGA obvodu. Adresa a dáta budú prenášané po spoločných vodičoch AD.

Konfiguračnú pamäť pre konfigurovanie FPGA obvod je tvorená obvodmi XC18V01 [19] a XCF01S [16]. Využívať a osadzovať sa však bude vždy len jedna z možností. Voľba závisí na dostupnosti súčiastky na trhu. Prepojenie pamäte s FPGA obvodom vychádza z doporučeného zapojenia uvedeného v dokumentácii príslušného FPGA obvodu (viz. 4.2.3). Pamäť je pripojená v režime slave, sériový prenos dát. Tento režim je nastavený pomocou vstupných pinov M0 - M1 na FPGA obvode, kde na piny M0, M1 je privedená logická nula a na pin M2 je privedená logická jednotka pomocou *pull-up* rezistoru. Konfiguračná pamäť spolu s FPGA obvodom sú zapojene v reťazci *boundary scan* a tak pre programovanie pamäte je možné použiť JTAG rozhranie. Toto rozhranie je vyvedené pomocou konektoru JP8 1x6 pinov. Zahájenie konfiguračného prenosu je riadené signálom RESET, ktorý je pripojený priamo z PCI zbernice. Zabezpečí sa tak, že po systémovom reštarte počítača bude FPGA obvod nahraný pomocou konfiguračnej pamäte. K signalizácii prenosu konfiguračného reťazca slúži dióda LED1. Dióda sa rozsvieti počas konfiguračného procesu a zhasína ak proces úspešne prebehol.

Obvodové zapojenie obsahuje štyri komunikačné kanály. Každý kanál je riadený mikroprocesorom Atmel AT89C2051, kde vstup tvoria signály zo strany FPGA obvodu a výstupom je sériové asynchrónne rozhranie UART. Zdrojom hodinového signálu pre procesory je kryštál o frekvencii 22 MHz. Tento kryštál sa vybudí pomocou procesoru U4 a výstupný hodinový signál je pripojený cez budič 74HC04 na ďalšie procesory U5 - U7. Žiadosť o prerušenie od procesoru na PCI zbernicu je signalizovaná pomocou LED3 - LED6. Procesory sa budú programovať externe a v doske budú osadené v pätici.

Výstupom dvoch komunikačných kanálov bude sériové asynchrónne rozhranie UART vyvedené na konektor XP4 a XP5. Tento konektor slúži na pripojenie externých modulov, ktoré prevádzajú rozhranie UART na iné typy rozhraní. Výstupný konektor obsahuje napájacie napätie, ktoré slúži k napájaniu externých modulov. Súčasťou rozhrania bude riadiaci signál DTR, ktorým sa bude riadiť smer komunikácie.

Zvyšné dva kanály sú zakončené rozhraním RS485 - galvanický neoddelene. Ako prevodník RS485 je použitý obvod LTC485, ktorý je zapojený pre obojsmernú komunikáciu. Smer komunikácie určuje signál DTR. Jumpery JP6 a JP7 určujú, že pri vysielaní na linku RS485 sú vysielné dáta privádzané aj na vstup. Vysielač tak *počuje*, čo vysiela. Táto možnosť sa dá zakázať nadstavením prepínačov na režim *bez echovania*. Výstup obvodu LTC485 je zakončený konektorom CANNON9. Rezistory R24, R25 (R14, R35) slúžia ako poistky pred prípadnou napäťovou špičkou. Zenervové diódy D7 - D10 (D11 - D14) slúžia k prapäťovej ochrane. Výstup rozhrania je zakončený jumpermi JP2 - JP5, ktoré slúžia k zakončeniu komunikačnej linky.

Obvodové zapojenie využíva tri rôzne napäťové úrovne: 5V, 3.3V a 2.5V. Napäťová úroveň 5V sa získava priamo z PCI zbernice a využíva sa pre napájanie mikroprocesorov. K stabilizácii napätia slúžia kondenzátory C21 - C26.

Napätie 3.3V sa využíva k napájaniu konfiguračných pamätí FPGA obvodu. Je privedené priamo zo zbernice a k stabilizácii slúžia kondenzátory C1 - C20.

Napätie 2.5V sa získava pomocou lineárneho pasívneho stabilizátora napätia REG1117A z 5V úrovne. Hodnoty kondenzátorov C51 a C52 sú stanovené v dokumentácii obvodu [4]. Toto napätie sa využíva k napájaniu jadra FPGA obvodu a je stabilizované pomocou keramických kondenzátorov C27 - C39.

Diódy LED7 - LED9 slúžia k oživeniu dosky a pre samotnú funkčnosť dosky nie sú potrebné.

5.3 Doska plošného spoja

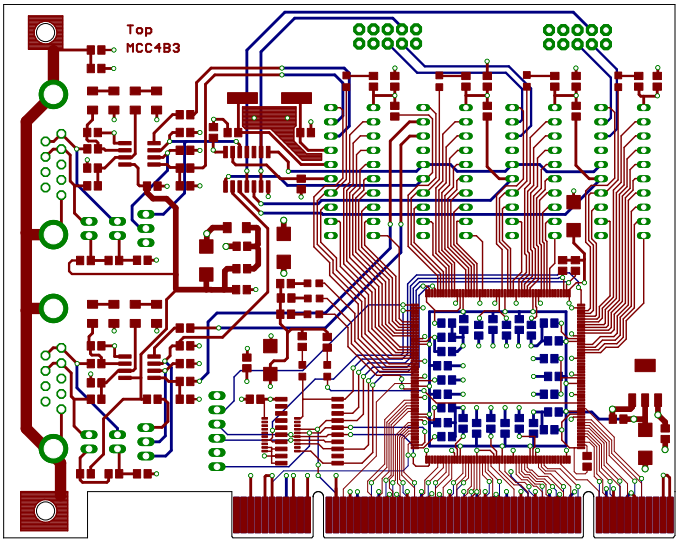
Na obrázku 5.1 je znázornená doska plošných spojov komunikačnej karty. Doska je tvorená štyrmi vrstvami, kde vrstvy *top* a *bottom* obsahujú signálové cesty a vo vnútorných dvoch vrstvách je po celej ploche dosky vyvedené napájanie a zem. Súčiastky sú osadené zo oboch strán, pričom na spodnej strany sú umiestené iba filtrovacie kondenzátory FPGA obvodu. Rozmer dosky je 120 x 95 mm čo umožňuje použitie karty v počítačoch PC. Spodná časť plošného spoja obsahuje konektor PCI zbernice. Tento konektor je vytvorený z medených plôch, ktoré sú vo finálnej realizácie pozlatené. Dosiahne sa tak lepšia mechanická odolnosť konektoru. Rozmery a rozmiestnenie pinov konektoru vychádza zo štandardu PCI zbernice.

Súčiastky sú na doske rozmiestnené do logických celkov. Znamená to, že obvody, ktoré spolu súvisia sú umiestnene čo najtesnejšie vedľa seba. FPGA obvod je umiestnený uprostred dosky v blízkosti PCI konektoru aby sa zachovalá maximálna definovaná dĺžka signálov (viz. 4.2.6). Tieto signály sú navrhnuté tak, aby ich dĺžky boli približne rovnaké. Zabráni sa tak oneskoreniam, ktoré by mohli vzniknúť na rôzne dlhých vodičoch. Konfiguračná pamäť spolu s konektorom JTAG sú umiestnené na ľavo od FPGA obvodu.

V hornej časti dosky sú osadené štyri pätice, ktoré slúžia pre osadenie mikroprocesorov. Procesory tak budú naprogramované externe mimo dosku. Zdroj hodinového signálu (kryštál) je umiestnený v blízkosti procesoru, aby sa zabránilo zkresleniu hodinového signálu.

Výstupy komunikačných kanálov sú vyvedené na okrajoch dosky aby boli ľahko dostupné. Konektory sériového rozhrania UART sú v hornej časti dosky a konektory CANNON9 sú umiestnené na ľavom okraji dosky a sú dostupné z vonkajšej strany počítača PC.

Na doske je rovnomerné rozmiestnených niekoľko tantalových kondenzátorov, ktoré zabezpečujú stabilizáciu napätia. Keramické kondenzátory sú umiestnené v blízkosti napájania integrovaných obvodov a zabráňujú tak zákmitom napájacieho napätia, ktoré vzniká v dôsledku rýchlych zmien v odbere napájacieho prúdu.



Obrázek 5.1: Doska plošných spojov (vrstva top a bottom).

Kapitola 6

Softvérové riešenie

Kapitola popisuje postup implementácie PCI radiča a obslužnej aplikácie. Proces implementácie bude vychádzať z návrhu uvedeného v kapitole 4.4. Navrhnutý projekt bol praktický realizovaný a postup konštrukcie a testovania je uvedený v závere práce 8.

6.1 Vývojové nástroje

Pre realizáciu PCI radiča boli použité vývojove nástroje: ModelSim a ISE. K implementácii bol použitý jazyk VHDL.

6.1.1 ModelSim

ModelSim je vývojový simulačný nástroj spoločnosti Mentor Graphics [5]. Tento program predstavuje základný stavebný prvok vývoja aplikácii pre FPGA obvody. Navrhnutú jazykovú konštrukciu je možne pomocou nástrojov ModelSimu simulovať a určiť tak jej chovanie v obvode. Súčasťou balička ModelSimu je množstvo knižníc, ktoré urýchľujú vývoj aplikácie. Pre realizáciu projektu bol použitý ModelSim XE III Starter 6.4b. Táto verzia je dostupná po registrácii na stránkach Xilinxu.

6.1.2 ISE

ISE (Integrated Synthesis Environment) je balíček vývojových nástrojov určený špeciálne pre FPGA obvody. Pomocou ISE je možné z danej jazykovej konštrukcie (jazyky VHDL alebo Verilog) vytvoriť konfiguračný reťazec pre FPGA obvody. ISE obsahuje množstvo ladiacich nástrojov a knižníc, ktoré môžu značne urýchliť vývoj aplikácie. Pre realizáciu projektu bol použitý ISE 10.1, ktorý je dostupný na stránkach Xilinxu.

6.2 Implementácia radiča PCI zbernice

Radič je rozdelený do niekoľkých funkčných blokov (viz. obrázok blokovej schémy 4.2.6):

- **Target**
- **Dekodér**
- **Blok parity**

- **Konfiguračný pamäť**
- **Pamäť zariadenia**

Každý blok realizuje niektorú z funkcii radiča a v jazyku VHDL bude implementácia bloku predstavovať VHDL komponentu. Všetky bloky budú popísane behaviorálne.

6.2.1 Target

Blok *target* je jadrom systému radiča PCI. Jeho úlohou je na základe požiadavkou na PCI zbernici riadiť činnosť ostatných závislých blokov v systéme. Tento blok je realizovaný pomocou stavového automatu a jeho implementácia sa bude mierne odlišovať od návrhu uvedeného v kapitole 4.4.

Stavový automat

Grafické znázornenie stavového automatu je uvedené v návrhovej časti 4.4.2. Automat je tvorený šesticou stavovou.

1. **IDLE** - počiatočný stav. Automat v tomto stave očakáva začiatok transakcie na zbernici. Automat do stavu IDLE prechádza vždy po systémovom reštarte alebo po dokončení transakcie.
2. **BUS_BUSY** - stav automatu, keď zariadenie detekuje na zbernici práve začatú transakciu. Zariadenie v tomto stave určí, komu patrí daná transakcia na zbernici a či sa jedná o čítanie alebo zápis.
3. **T_AROUND** - stav automatu, keď sa na zbernici vykonáva vratný cyklus. Znamená to, že kontrolu nad dátovými signálmi získava cieľové zariadenie. V ďalšom stave cieľové zariadenie vystaví platné dáta na zbernici.
4. **DATA** - stav automatu, keď sú na dátovej zbernici PCI vystavené platné dáta. Je potrebné rozlíšiť, či sa jedna o čítanie alebo zápis. V prípade zápisu cieľové zariadenie číta dáta zo zbernice. V prípade transakcie čítania zariadenie zapisuje dáta na zbernicu.
5. **DISCONNECTED** - koncový stav automatu, v ktorom sú signály na PCI zbernici deaktivované ale stále budené cieľovým zariadením.
6. **REL_BUS** - Neočakávaný koncový stavu automatu. Môže nastať v prípade nekonzistentných dát na zbernici. Prenos je ukončený a automat prechádza do počiatočného stavu IDLE.

Uvedený stavový automat je Mealyho automat. Znamená to, že na základe vstupných podmienok sa zmení stav automatu a sú nadstavené výstupy podmienky. Nasledujúci výklad popisuje prechodové funkcie automatu a funkčnosť jednotlivých signálov.

IDLE:

goto IDLE	if FRAME# = 1
goto BUS_BUSY	if FRAME# = 0

BUS_BUSY:

goto IDLE	if Hit_memory = 0 or Hit_config = 0
goto BUS_BUSY	if FRAME# = 0 and Hit_memory = 0 and Hit_config = 0
goto DATA	if (Hit_memory = 1 or Hit_config = 1) and write = 1
goto T_AROUND	if (Hit_memory = 1 or Hit_config = 1) and read = 1

DATA:

goto DISCONNECT	if IRDY# = 0 and FRAME# = 1
goto DATA	if IRDY# = 1 and FRAME# = 0
goto REL_BUS	if IRDY# = 1 and FRAME# = 1

T_AROUND:
goto DATA

DISCONNECT:
goto IDLE

REL_BUS:
goto IDLE:

Popis činnosti stavového automatu

Nasledujúci výklad popisuje činnosť automatu a nastavenie výstupných signálov na PCI zbernici.

IDLE - v stave IDLE stavový automat vzorkuje hodnotu signálu FRAME#. Ak detekuje aktívny signál FRAME#, automat prechádza do stavu BUS_BUSY. Výstupné signály v stave IDLE nezasahujú do komunikácie na zbernici a sú nastavené do stavu vysokej impedancie (Z).

DEVSEL#: Z

TRDY#: Z

STOP#: Z

BUS_BUSY - v tomto stave automat vzorkuje hodnoty signálov Hit_config a Hit_memory. Na ich základe rozhoduje či daná transakcia patrí práve jemu cieľovému zariadeniu alebo inému zariadeniu na zbernici. Ak sa transakcia týka daného cieľového zariadenia, tak na základe internej logiky je nastavený nový stav automatu a aktivovaný signál DEVSEL#. Signály tak budú mať nasledujúce hodnoty:

DEVSEL#: 0

TRDY#: 1

STOP#: 1

T_AROUND - automat zo stavu T_AROUND automatický prechádza do nového stavu DATA. Výstupne signály sa nemenia a ich nadstavenie závisí na predchodzom stave automatu.

DATA - v stave DATA cieľové zariadenie aktivuje signál TRDY# a informuje tak iniciátora, že je pripravené pristupovať k dátovej zbernici. V prípade jednoduchých dátových prenosov, je potrebné aktivovať výstupný signál STOP#. Signály tak budú mať nasledujúce hodnoty:

DEVSEL#: 0
TRDY#: 0
STOP#: 0

Ak sa jedná o zápisovú transakciu, cieľové zariadenie pristupuje k obsahu zbernice AD a ukladá ho do internej logiky. V prípade transakcie čítania cieľové zariadenie získava kontrolu nad zbernicou a zapisuje nové dáta na zbernicu. Ak je zariadenie pripravené vyčítať alebo zapísať dáta na zbernicu musí byť aktivovaný signál TRDY#.

DISCONNECTED - v koncovom stave DISCONNECTED cieľové zariadenia deaktivuje signály DEVSEL#, TRDY# a STOP#. Deje sa tak nadstavením signálov na hodnoty:

DEVSEL#: 1
TRDY#: 1
STOP#: 1

Automat prechádza do ďalšieho stavu IDLE automatický bez vstupných podmienok.

REL_BUS - v stav REL_BUS sú deaktívované signály DEVSEL#, TRDY# a STOP#. Automat prechádza do ďalšieho stavu IDLE automatický bez vstupných podmienok.

Funkčnosť riadiacích signálov

Primárnou úlohou bloku *target* je na základe požiadavkov na PCI zbernici riadiť činnosť ostatných závislých blokov v systéme a priebeh komunikácie na PCI zbernici. So svojím okolím je tento blok prepojený pomocou vstupných a výstupných signálov, ktoré sa podieľajú na riadiacej činnosti.

S PCI zbernicou je prepojený pomocou signálov RST, C/BE#, CLK, FRAME#, IRDY#, TDRY#, DEVSEL# a STOP#.

Signály RST, C/BE#, CLK, IRDY# a FRAME# sú určene len pre čítanie a sú privedené priamo zo zbernice do vnútornej logiky. Tieto signály predstavujú vstupne podmienky implementovaného stavového automatu.

Signály TRDY#, DEVSEL# a STOP# sú na základe stavov automatu nastavované na príslušné hodnoty (viz. 6.2.1) a riadia tak komunikačnú činnosť na PCI zbernici. PCI zbernica je zdieľaná zbernica a je potrebné zabezpečiť, aby tieto signály nebudili zbernicu ak sa cieľové zariadenie nezúčastňuje komunikácie. Deje sa tak použitím vstupno-výstupných bufferov (IOB) na ich výstupe . Činnosť týchto IOB je riadená pomocou signálu EN_STROBE.

Signály pre riadenie internej logiky sú implementované nádsledovne:

Popis riadiacich signálov

1. **Hit_Config a Hit_Memory** - jednobitové vstupné signály, ktoré sú riadené blokom *dekóderu*. Aktivovaním signál Hit_Memory sa určuje, že zariadenie rozpoznalo svoju adresu počas adresnej fázy na zbernici a transakcia tak patrí jemu. Funkčnosť Hit_Config je rovnaká ako Hit_Memory a týka sa výlučne konfiguračných prenosov. Signály sú aktívne v hodnote logickej 1.

2. **Decode** - jednobitový signál, ktorý určuje, že blok Dekodéru môže dekódovať adresu vystavenú na zbernici. Určuje tak, že na zbernici práve prebieha adresná fáza. Signál je aktivovaný v hodnote logickej 1.

3. **Do_Operation** - jednobitový signál prepojený s blokom *dekóderu*. Signál určuje, že na zbernici práve prebieha dátová fáza. Ak je signál aktívny (logická 1), dekodér prečíta platne dáta zo zbernice. Ak sa jedná o čítanie, *dekodér* vystaví platné dáta na zbernici.

4. **OperationType** - množina štyroch signálov, ktoré určujú aký typ transakcie prebieha na zbernici (viz. 3.6). Hodnota signálov je závislá na bitoch C/BE#.

5. **Enable_Parity** - jednobitový signál prepojený s blokom pre výpočet parity. Nadstavením signálu na logickú hodnotu 1 sa určí, že cieľové zariadenie vypočíta paritu na zbernici a nastaví signál parity PAR.

6. **En_Strobe** - jednobitový signálov, ktorý riadi trojstavový budič signálov TRDY#, STOP# a DEVSEL#. Ak transakcia patrí cieľovému zariadeniu, tak je nastavený na hodnotu logickej 0 a určí sa tým, že signály TRDY#, STOP# a DEVSEL# budú budiť PCI zbernicu. V opačnom prípade budú tieto signály nastavané do stavu vysokej impedancie a nezúčastnia sa komunikácii na zbernici.

7. **En_StrobeAddress** - jednobitový signál, ktorý riadi trojstavový budič na dátových vodičoch PCI zbernice. Ak je nastavený na hodnotu logickej 0, znamená to, že zariadenie môže vystaviť platné dáta na zbernicu. V opačnom prípade zariadenie vyčíta hodnoty signálov na dátovej zbernici.

8. **En_StrobeParity** - jednobitový signál, ktorý riadi smer komunikácie signálu PAR. Ak sa jedná o transakciu čítanie z cieľového zariadenia, signál je nastavený na hodnotu logickej 0, čo znamená, že signál PAR bude riadený cieľovým zariadením.

Stavový automat je v jazyku VHDL implementovaný behaviorálne. Jeho činnosť je riadená hodinovým signálom CLK, ktorý je privedený zo zbernice PCI. Na každú nábežnú hranu hodinového signálu sú vzorkované všetky vstupné signály do komponenty a na základe ich hodnôt automat prechádza jednotlivými stavmi. Signál RESET zabezpečuje, že v prípade systémového reštartu automat prechádza do počiatočného stavu a všetky výstupne signály sú odpojene od PCI zbernice.

Súčasťou komponenty *target* je ďalej logika, ktorá vyhodnocuje typ transakcie. Zariadenie rozlišuje štyri typy transakcii: pamäťové čítanie, pamäťový zápis, konfiguračné čítanie a konfiguračný zápis.

6.2.2 Dekóder

Blok Dekóder v systéme radiča PCI splňuje niekoľko funkcii nevyhnutných pre správnu činnosť radiča zbernice. S PCI zbernicou je prepojený pomocou AD signálov. Pre riadenie jeho funkčnosti sa využívajú signály RST, CLK, IDSEL# a C/BE# z pohľadu PCI zbernice. Signály DECODE, DO _OPERATION a OPERATION _TYPE z pohľadu vnútornej logiky. K bloku *dekóder* sú ďalej pripojené bloky *konfiguračná pamäť a pamäť zariadenia*. *Dekóder* na základe požiadavkou na PCI zbernici pristupuje do týchto pamäti a vykonáva potrebné operácie. Nasledujúci výklad postupne popisuje jednotlivé funkcie *dekóderu*.

Proces dekódovania

Proces dekódovania transakcie môže prebiehať v dvoch rôznych variantach. Prvá varianta prebieha, ak sa jedná o pamäťové prenosy a druhá ak sa jedná o konfiguračné prenosy. V prípade pamäťového prenosu *dekóder* porovnáva bázovú adresu s adresou vystavenou na zbernici. Bázová adresa je uložená v konfiguračnej pamäti zariadenia a v pomocnom registre. Hodnota v pomocnom registre sa využíva pre rýchlejšie porovnanie adresy vystavenej na zbernici s hodnotou bázovej adresy. Je tak možné dosiahnuť dekódovanie adresy na úrovni médium DEVSEL#. Ak *dekóder* zistí, že adresa na zbernici patrí cieľovému zariadeniu nadstaví signál Hit_Memory na hodnotu logickej 1 a uloží adresu do registra.

V prípade konfiguračných prenosov *dekóder* číta hodnotu signálu IDSEL#. Ak zistí, že je signál aktívny spoločne so signálom DO_DECODE (signál riadený blokom *target*) nastaví signál Hit_Config na hodnotu logickej 1 a uloží hodnotu adresy do registra.

Proces zápisu dát

Zápis dát je proces, keď *dekóder* číta dáta z PCI zbernice a zapisuje ich do pamäti zariadenia. V princípe *dekóder* rozlišuje dva typy pamäťových prístupov. Jeden je do konfiguračnej pamäte a druhý do pamäte zariadenia. Typ operácie je určený signálom OPERATION_TYPE. Samotný proces zápisu je podmienený signálom DO_OPERATON. Ak je signál aktívny (logická 1), *dekóder* vykonáva proces zápisu.

Ak sa jedná o konfiguračný zápis, *dekóder* nastaví na výstupnom rozhraní (rozhranie s konfiguračnou pamäťou) platnú adresu (adresa je uložená v registre z procesu dekódovania transakcie) a riadiace signály pre blok konfiguračnej pamäte. *Dekóder* rozlišuje pamäťové bunky konfiguračnej pamäti, ktoré sú určene iba pre čítanie. Do týchto buniek nie sú povolené zápisy. V prípade BAR registra je umožnený zápis iba do horných 24 bitov. Je tak zabezpečená alokácia pamäte o veľkosti 4kB. Počas konfiguračného prenosu sú vždy platné všetky bajty dátovej časti, preto prístupy do konfiguračnej pamäte sú výlučne 32 bitov široké.

V prípade pamäťového zápisu Dekóder nadstaví na výstupnom rozhraní platnú adresu a riadiace signály. Pamäť zariadenia je 8 bitov široká, Dekóder tak musí vybrať platné dáta a nadstaviť ich na dátovú zbernicu pamäte zariadenia.

Proces čítania dát

Čítanie dát je proces, keď *dekóder* vyčíta obsah pamäte zariadenia na danej adrese a vystaví ho na výstupnom rozhraní (PCI zbernici). Podobne ako u zápisu dát, čítanie rozlišuje dva typy pamäťových prístupov: čítanie z konfiguračnej pamäte a čítanie z pamäte zariadenia. Typ operácie je určený signálom OPERATON_TYPE a beh procesu je riadený signálom DO_OPERATION.

Proces čítanie dát zaberie dva hodinové takty. V prvom hodinovom takte sú nastavené riadiace signály a adresa príslušnej pamäte. V druhom hodinovom takte *dekóder* vyčíta dáta z pamäte a vystaví ich na zbernicu PCI.

Komponenta *dekóder* je tvorená niekoľkými procesmi a registrami pre uloženie hodnôt. Procesy *read_data* a *write_data* zabezpečujú prepínanie dát medzi zbernicou PCI a internou logikou. Pomocou procesu *setup* komponenta zabezpečuje nastavenie riadiacich signálov pre jednotlivé bloky pamätí (blok konfiguračnej pamäte a blok pamäte zariadenia) na základe typu transakcie. Register *base_address* slúži k uloženiu bázovej adresy zariadenia. Táto hodnota je využívaná v procese *compare_address* pre rýchle porovnanie počas adresnej

fázy. Beh všetkých procesov je synchronizovaný pomocou signálu CLK, ktorý je vyvedený priamo zo zbernice PCI.

6.2.3 Blok parity

Blok parity zabezpečuje nasstavenie príslušnej parity (signálu PAR) počas transakcii čítania. Výpočet parity prebieha na signáloch AD a C/BE#. Na základe ich hodnôt je výstupný signál PAR nadstavený tak, aby tieto signály spolu so signálom PAR obsahovali párny počet logických hodnôt 1. Pre výpočet parity je použitá funkcia XOR. Tento blok je riadený pomocou signálu PARITY_ENABLE. Ak je signál nastavení na hodnotu logickej 1 prebieha výpočet parity. V opačnom prípade sa parita nepočíta a signál PAR je nastavený do stavu vysokej impedancie.

Parita je počítaná pomocou funkcie XOR, ktorá ma na vstupe 36 signálov. Niektoré nástroje môžu mať problémy so syntézou 36 vstupového XORu a je preto potrebné skontrolovať proces syntézy, aby bol XOR správne vytvorený.

6.2.4 Konfiguračná pamäť

Konfiguračná pamäť predstavuje blok pamäte, ktorého štruktúra je definovaná štandardom PCI (viz 3.4). K realizácii pamäti je použite pole o veľkosti 64x32 bitov. V tomto priestore sú implementované základne registre, ktoré sú nevyhnutné pre činnosť zariadenia. Pamäť je pripojená k bloku *dekóder* pomocou riadiacich a dátových signálov. Riadiace signály sú CLK, WEA a ENABLE. Signál CLK je vyvedený priamo s PCI zbernice a zabezpečuje synchronizáciu prístupov do pamäti s PCI zbernicou. Signál WEA riadi typ prístupu do pamäti. Ak je nastavený na hodnotu logickej 1, jedná sa o zápis do pamäte. V opačnom prípade ide o čítanie z pamäte. Signál ENABLE povoľuje prístup do pamäte. Nastavením tohto signálu do logickej 0 sa nevykonáva žiadna operácia. Adresa pre konfiguračnú pamät je široká 6 bitov a adresuje tak $2^6 = 64$ doublwordov. Dátová zbernica je široká 32 bitov.

Po systémovom reštarte sa konfiguračná pamäť dostáva do počiatočného stavu tz. že niektoré pamäťové bunky sú nastavené na konštante hodnoty:

1. **ID výrobcu** - 1022 Hexa (AMD)
2. **ID zariadenia** - 0001 Hexa
3. **Status register** - 0200 Hexa (medium DEVSEL)
4. **Class code** - 07 Hexa (hodnota určuje, že ide o zariadenie komunikačný kontrolór)
5. **BAR** - spodných 12 bitov bude počas každej transakcie nadstavených na nulu. Zabezpečí sa tak alokovane 4kB miesta pamäte.

Pamäť je navrhnutá v jazyku VHDL tak, aby proces syntézy rozpoznal a využil Block RAM pamäť obvodu FPGA.

6.2.5 Pamäť zariadenia

Blok pamäte zariadenia je rozdelený na päť časti. Prvé štyri časti predstavujú pamäť pre komunikačné procesy a poslednú časť tvoria riadiace registre zariadenia. Blok pamäte (všetkých päť časti) má dátovú šírku 8 bitov a rozprestiera sa v priestore 4096 bajtov. Nie celá pamäť je využitá pre uloženie dát. V kapitole návrh systému (viz 4) je uvedené rozloženie

adresového priestoru zariadenia. Pre každý komunikačný kanál tak pripadá veľkosť pamäte 256 bajtov s rozostupmi jednotlivých kanálov po 256 bajtov. Pre tento dôvod sú pre realizáciu komunikačných kanálov využité blokové RAM pamäte FPGA obvodu o veľkosti 512 bajtov. Využije sa tak vždy len polovica blokovej RAM pamäte. Vnútorná logika zariadenia musí na základe adresy rozlišovať do ktorej blokovej pamäte sa bude pristupovať. Blok pamäte je s *dekóderom* prepojený pomocou 12 bitov širokej adresovej zbernice ($2^{12} = 4096$) a 8 bitov širokej dátovej zbernice a riadiacimi signálmi WEA, ENABLE a CLK.

Výstupom bloku pamäte zariadenia sú štyri komunikačné kanály, ktoré sú pripojené priamo na mikroprocesory. Komunikácia medzi mikroprocesorom a pamäťou zariadenia zabezpečuje vnútorná logika. Úlohou tejto logiky je na základe požiadavkov od procesora pristupovať do blokovej RAM pamäte a vykonáva príslušne operácie. Mikroprocesor je s blokom pamäte prepojený pomocou signálov AD[7:0], OEA, WEA, CEA a RST/. Signály OEA, WEA a CEA sú riadiace signály a ich priebeh je uvedený v dokumentácii [2]. Signál RST slúži k reštartovaniu komunikačných procesov a je riadený registrom CTRL1. Adresa a dáta sú multiplexované na spoločné vodiče a na ich výstupe sú umiestnene IOB aby komunikácia mohla prebiehať obojsmerne.

6.2.6 Výsledky implementácie

Výsledná implementácia radiča je rozdelená do piatych komponent, kde každá komponenta realizuje jednu z funkcii radiča a je prepojená s ostatnými blokmi pomocou vstupno-výstupných signálov. Výsledným riešením tak bude jedná nadradená komponenta PCI_top, ktorá definuje prepojenie všetkých komponent v systéme a ich vzájomné závislosti. V jazyku VHDL je táto komponenta definovaná výlučne štrukturálne.

Takto vytvorenú komponentu je možné pomocou nástrojov ISE nahrať do FPGA obvodu a otestovať jej správnu funkčnosť. Samotný proces nahrávania je posledným krokom a predchádza mu niekoľko iných krokov:

1. **Simulacia**
2. **Syntéza**
3. **Implementácia dizajnu**
4. **Nahrávanie**

Všetky vyššie uvedené kroky je možné vykonať pomocou nástrojov ModelSim a ISE.

Simulácia

K simulácii implementovanej jazykovej konštrukcie slúži simulačný nástroj ModelSim. Pomocou tohto nástroja je možne výsledný dizajn skompilovať (odhaliť syntaktické chyby) a odsimulovať (odhaliť sémantické chyby). ModelSim umožňuje simulovať každú komponentu zvlášť alebo všetky spoločne, ako jeden funkčný celok. Proces simulácie vyžaduje navrhnúť testbench súbory, ktoré simulujú okolie modelovaného zariadenia. Výsledok simulácie je graf reprezentovaný pomocou priebehov.

K simuláciu radiča zbernice bol navrhnutý testbench súbor, ktorý simuluje výslednú komponentu PCI_top zo strany PCI zbernice. Tento testbench simuluje transakcie pridelenie adresy cieľovému zariadeniu a následne pamäťové čítanie a zápis. V prílohe na obrázku B je uvedená časť tohto priebehu. Jedná sa o konfiguračný zápis a následne konfiguračné čítanie.

Na obrázku je vidieť, že v prvom kroku (konfiguračný zápis) sú do BAR registru (adresa 0x10000) zapísane hodnoty 0xFFFFFFFF a následne je hodnota BAR registru vyčítaná. Zariadenie vracia hodnotu 0xFFFFF000, pretože sa jedná o čítanie z BAR registru, kde spodných 12 bitov je pevne nadstavených a určuje koľko pamäte má byť alokovaných pre zariadenie.

Signály uvedené modrou farbou znázorňujú stav vysokej impedancie a signály uvedené červenou farbou znázorňujú neinicializovanú hodnotu. Simulácia pomocou testbench súborov predstavuje ideálne chovanie hardvéru a často krát sa môže líšiť od reality.

Syntéza

Syntéza predstavuje proces, keď z výslednej implementácie (komponenta PCI_top) je vytvorený NetList prvkov cieľovej architektúry (NGO súbor). Vstupom syntézy sú implementované komponenty, knižnice prvkov cieľovej architektúry a užívateľom definované obmedzenia tzv. constrains. Pomocou constrains je možne definovať rôzne omedujúce podmienky. Pre radič PCI budú použité obmedzenia na minimálnu frekvencia, na ktorej obvod musí pracovať, napäťový štandard na vstupných pinoch a priradenie vstupno-výstupnych pinov danej komponente. Proces syntézy určí čí daná implementácia bude splňovať časové podmienky a koľko systémových prostriedkov zaberie výsledný dizajn v FPGA obvode.

Pre radič PCI zbernice bol časová obmedzujúca podmienka nadstavená na hodnotu 33 MHz a komponenta PCI_top bola na mapovaná na piny podľa špecifikácie obvodovej schémy.

Výstupom syntézy sú nasledujúce hodnoty:

Minimum period: 14.147ns (Maximum Frequency: 70.686MHz)
Minimum input arrival time before clock: 7.831ns
Maximum output required time after clock: 11.558ns

Number of Slices:	269	out of	768	35%
Number of Slice Flip Flops:	121	out of	1536	7%
Number of 4 input LUTs:	503	out of	1536	32%
Number used as logic:	375			
Number used as RAMs:	128			
Number of IOs:	49			
Number of bonded IOBs:	49	out of	140	35%
Number of BRAMs:	4	out of	8	50%
Number of GCLKs:	1	out of	4	25%

Implementácia dizajnu

Implementácia dizajnu je proces, keď NetList prvkov cieľovej architektúry je na mapovaný do vnútornej logiky FPGA obvodu. Proces prebieha automatický a v prípade potreby je možne vykonávať ručnú modifikáciu.

Nahrávanie

Nahrávanie je posledným krokom pred spustením aplikácie. Pomocou nástrojov ISE je umožnené na základe výsledkov syntézy a implementácie dizajnu vygenerovať výsledný

binárny súbor určený pre FPGA obvod alebo pre konfiguračnú pamäť. Počas generovania binárneho súboru pre konfiguračnú pamäť je potrebné správne zvoliť typ pamäte a jej puzdro. V prípade komunikačnej karty ide o pamäť o veľkosti XC18V01 s puzdrom SO20.

Pred samotným procesom nahrávania je potrebné ku komunikačnej karte pripojiť JTAG programátor (Platform Cable USB II). Pripojenie pinov viz. tabuľka 6.1. Obvody sú zapojené v reťazci boundary scan a je tak možne programovať konfiguračnú pamäť a samotný FPGA obvod. Pre správne fungovanie radiča PCI je potrebné naprogramovať konfiguračnú pamäť. Zabezpečí sa tak, že po každom reštarte alebo spustení bude nahraný dizajn z konfiguračnej pamäte do FPGA obvodu.

Pinheader JP8	**Programátor**
1	VREF
2	TDO
3	TMS
4	TDI
5	GND
6	TCK

Tabulka 6.1: Pripojenie JTAG programátora.

Kapitola 7

Ovládač zariadenia

Ovládač zariadenia je program, ktorý realizuje komunikačnú vrstvu medzi operačným systémom a hardvérom. Obecne sa jedná o rozhranie, ktoré na základe príkazov z vyšších vrstiev (aplikačný program, operačný systém) realizuje sériu príkazov na zbernici, pomocou ktorých komunikuje priamo s cieľovým zariadením. Ovládač zariadenia umožňuje vytvárať aplikácie nezávisle na špecifikácii hardvéru. Znamená to, že ovládač môže mať zo strany aplikácie jednotné rozhranie pre rôzne typy zariadenia.

Z pohľadu implementácie ovládače zariadenia ich môžeme rozdeliť do dvoch základných skupín podľa toho, v akej časti systému pracujú:

Kernel-mode (ring 0) - Ovládač používaný v kernel-mode je charakteristický lepšou výkonnosťou. Avšak prípadná chyba ovládača môže spôsobiť pád systému. Rozhranie ovládača je sprístupnené pomocou špeciálnych funkcii, ktoré sú súčasťou jadra systému.

User-mode (ring 3) - Ovládače používane v user-mode sú charakteristické lepšou stabilitou. Prístup k zariadeniam je uskutočnený pomocou funkcii jadra systému a systém tak môže riadiť priebeh komunikácie.

S pohľadu platformy x86 je možne rozdeliť ovládače na dve základné skupiny:

1. **Windows** - ovládač určený pre operačný systém Windows. Tento typ ovládača je založený na platforme Windows a môžu bežať v dvoch rôznych módoch: kernel-mode a user-mode. V súčasnosti sa ovládače pre systém Windows využívajú výhradne v user-mode. Zabráni sa tak pádu systému v prípade chybne navrhnutom ovládači.

2. **Unix (Linux)** - ovládač založený na platforme Unixových systémov pracujú výhradne len v kernel-mode systému. Prístup k ovládačom z užívateľskej aplikácie je tak pomocou špeciálnych systémových funkcii jadra systému. V systéme Unix je ovládač súčasťou jadra a jeho načítanie vyžaduje opätovné načítanie systému. V systéme Linuxe sú ovládače dynamický načítane pomocou modulov a môžu byť pridávane a odoberané počas behu systému.

7.1 Implementácia ovládača

Pre realizáciu ovládačov zariadenia existuje množstvo nástrojov. K implementácii ovládača je použitý nástroj Jungo WinDriver a Device Driver Kit. Ovládač implementovaný pomocou

Jungo WinDriver bol použitý k ladeniu a testovaniu radiča PCI zbernice. Jedná sa o jednoduchý ovládač, ktorý realizujú základné komunikačné prvky s cieľovým zariadením. Ovládač implementovaný pomocou DDK predstavuje komplexný ovládač pre obsluhu komunikačnej karty. Tento ovládač bol implementovaný zadávateľom projektu a jeho implementácia nie je súčasťou diplomovej práce.

7.1.1 Testovacie ovládače

Ovládače určene pre testovanie a ladenie radiča PCI zbernice boli vyvinuté pomocou nástroju Jungo WinDriver. Tento nástroj umožňuje jednoduchým spôsobom pomocou grafického užívateľského rozhrania vytvoriť funkčný ovládač zariadenia pre systémy Windows a Linux. Súčasťou vygenerovaného ovládača bude obslužná aplikácia, pomocou ktorej je možne vytvorený ovládač otestovať. Nasledujúci výklad popisuje vytvorenie ovládača, ktorý podporuje základe operácie na danom zariadení. Jedná sa o operácie konfiguračne a pamäťové čítanie a zápis.

Vytvorenie ovládača

Ovládač bol vytvorený pomocou sprievodcu. Proces prebiehal v niekoľkých korkoch:

1. V prvom kroku softvér Jungo WinDrive detekuje všetky pripojené zariadenia na PCI zbernici. Karta PCI je v systéme zariadení definovaná s parametrami ID výrobcu rovné 1022 (AMD) a ID zariadenia rovné 1. V tomto kroku je vygenerovaný súbor *.inf.

2. V ďalšom kroku je načítaná pamäť zariadenia a konfiguračný priestor. V tomto kroku je potrebné definovať registre, pamäťové miesta a prerušenia, ku ktorým chceme aby testovací softvér mohol pristupovať.

3. Posledným krokom je generovanie kódu. Generovanie je plné automatické. Je potrebné vybrať správnu architektúru PC (x86) jazyk v ktorom má byť ovládač vygenerovaný a vývojový nastroj, pomocou ktorého je možne ovládač editovať a skompilovať. K realizácii ovládača bol použitý jazyk ANSI C a vývojový nástroj Visual Studio 2008.

Vygenerovaný obslužný ovládač je spojený s testovacou aplikáciou, pomocou ktorej je možné pristupovať do definovaných registrov a pamäťových miest (definované boli v 2. kroku sprievodcu). Štruktúra ovládač je tvorná niekoľkými súbormi. Súbory wdc_diag_lib.c a diag_lib.c obsahujú obslužné rutiny, ktoré pristupujú do nižších vrstiev jadra systému. Súbory test_diag.c a test_lib.c slúžia k implementácii obslužnej aplikácie.

7.1.2 Ovladač zariadenia

Štruktúra ovládača zariadenia je uvedená na obrázku v prílohe C. Ovládač je tvorený knižnicou mscom32.dll, ktorá poskytuje jednotné komunikačné rozhranie API pre užívateľov aplikácie, bez ohľadu na typ komunikačnej linky. Táto knižnica beží na ringu 3 a využíva funkcie knižnice mscom.sys, ktorá beží na ringu 0. Knižnica mscom.sys sprostredkúva služby ovládača dosky pomocou súboru PciMccNT.inf. Súčasťou ovládača je podpora starších operačných systémov Windows. V prílohe na CD sú uvedené zdrojové súbory implementovaného ovládača.

Kapitola 8

Testovanie

Kapitola popisuje proces osadenia, oživenia a testovanie komunikačnej karty PCI.

8.1 Osadenie dosky

V prílohe na obrázku D je uvedené rozmiestnenie súčiastok na doske plošného spoja. Vyrobená doska bola osadená pomocou osadzovacieho automatu.

8.2 Oživenie komunikačnej karty

K procesu oživenia dosky je potrebná mať k dispozícii testovací počítač. Kartu je potrebné osadiť do voľného slotu PCI zbernice. Po spustení počítača by sa mala na okamih rozsvietiť červená LED dióda, ktorá signalizuje nahrávanie konfiguračného reťazca do FPGA obvodu. Jej zhasnutím FPGA indikuje, že konfiguračný reťazec bol úspešne nahraný do obvodu. V závislosti na type PC, niekoľko sekúnd po štarte systému a úspešnom konfiguračnom procese systémový softvér pridelí bázovú adresu všetkým zariadeniam na zberniciach. Ak bázová adresa bude úspešne pridelená na doske sa rozsvietiť zelená LED dióda. Tento proces je signalizovaný aj krátkym pípnutím systémového reproduktoru. Ak oživenie dosky prebehlo úspešne, prítomnosť dosky by mala byť indikovaná systémovým softvérom počas zavádzania systému (ID venor 0x1022 a ID device 0x1).

8.3 Testovanie komunikačnej karty

Test funkčnosti karty bol overený pomocou testovacej aplikácie v prostredí Microsoft Windows XP. Pred samotným spustením testu je potrebné nainštalovať ovládač zariadenia, ktorý je dostupný na priloženom CD. Po spustení testovacej aplikácie (súčasť CD) sa na obrazovke objaví obsah konfiguračnej pamäte. Hodnoty by sa mali zhodovať s nadstavenými konštantami konfiguračnej pamäte. V uvedenom príklade viz fotodokumentácia na priloženom CD je vidieť, že karta bola osadená na zbernici 0x00, slote 0xB a pridelená bázová adresa bola 0xE9000000 o veľkosti 0x1000. Aplikácia ponúka niekoľko ďalších možnosti testovania karty. Jedná sa o prácu s konfiguračným priestorom. Všetky tieto možnosti boli vyskúšané a overila sa tak základná funkčnosť komunikačnej karty, ktorá prebehla bez problémov. K testovaniu funkčnosti jednotlivých komunikačných kanálov je potrebné kartu pripojiť k rezaciemu stolu.

8.4 Ďalší vývoj

Súčasné FPGA obvody ponúkajú veľký výpočtový výkon a množstvo systémových prostriedkov preto by sa mohol ďalší vývoj uberať niekoľkými smermi:

- implementovať logiku procesorov do obvodu FPGA
- použiť zbernicu typu PCI-Express.

Kapitola 9

Záver

Cieľom tejto práce bolo vyhotoviť komunikačnú kartu pre riadenie rezacích strojov. Realizovaná karta je určená pre počítače PC a zasúva sa do voľného slotu PCI zbernice. Obsahuje štyri komunikačné kanály, ktoré komunikujú s osami rezacích strojov pomocou rozhrania RS485.

Hlavným stavebným prvkom komunikačnej karty je FPGA obvod Spartan II. Tento obvod riadi činnosť na PCI zbernici a realizuje pamäť pre komunikačné procesy. Funkčnosť FPGA obvodu určuje softvér napísaný vo VHDL jazyku. Karta obsahuje štyri komunikačné kanály, ktoré sú realizovane 8-bitovými mikroprocesormi Atmel. Tieto mikroprocesory pristupujú do pamäte FPGA obvodu, kde každý kanál má vyhradený adresový priestor, do ktorého môže mikroprocesor pristupovať. Dva komunikačné kanály sú zakončené rozhraním RS485 - galvanický neoddelene. Zvyšne dva kanály sú zakončené sériovým asynchrónnym rozhraním, ktoré slúži pre pripojenie externých modulov.

Pri realizácii komunikačnej karty sa jednalo o využitie hardvérových a softvérových znalostí. Bolo tak potrebné uplatniť postupy Hardware/Software Codesignu, aby jednotlivé časti správne spolupracovali. Tento projekt popisuje návrh od samotného počiatku, ktorý začína návrhom a analýzou funkcií zariadenia, výberom platformy, voľbou správnych súčiastok, návrhom a vyhotovením dosky a končí implementáciou softvéru.

Výsledkom projektu bola vyhotovená funkčná komunikačná karta. V spojení s ovládačom a príslušným softvérom (implementácia softvéru nie je súčasť tejto práce) karta predstavuje komunikačný prvok, ktorý bude zaradený v priemyselnom odvetí a bude slúžiť k ovládaniu servo motorov rezacích strojov. Predpokladané množstvo vyrobených kusov bude 500 ročne.

V prílohe E a na priloženom CD sú uvedené fotografie komunikačnej karty PCI.

Literatura

[1] CadSoft. [Online; navštíveno 20.4.2010].
URL http://www.cadsoft.de/

[2] Dokumentacia obovodu DS1609S. [Online; navštíveno 13.1.2010].
URL http://datasheets.maxim-ic.com/en/ds/DS1609.pdf

[3] Dokumentacia obovodu LTC485. [Online; navštíveno 10.11.2009].
URL http://cds.linear.com/docs/Datasheet/485fh.pdf

[4] Dokumentacia obovodu REG1117A. [Online; navštíveno 10.11.2009].
URL http://focus.ti.com/lit/ds/symlink/reg1117.pdf

[5] Mentor Graphics. [Online; navštíveno 3.5.2010].
URL http://www.mentor.com/

[6] Open Cores. [Online; navštíveno 2.2.2010].
URL http://www.opencores.com/

[7] PCI Local bus specification Revision 2.2. [Online; navštíveno 20.9.2009].
URL http://www.pcsig.com/

[8] Serial and UART Tutorial. [Online; navštíveno 5.4.2010].
URL http://www.freebsd.org/doc/en/articles/serial-uart/

[9] Atmel: Dakumentácia obovdou AT89C2051. [Online; navštíveno 13.4.2010].
URL http://www.atmel.com/dyn/resources/prod_documents/doc0368.pdf

[10] HW.CZ: RS 485. [Online; navštíveno 5.4.2010].
URL http://hw.cz/docs/rs485/rs485.html

[11] Kořenek, J.; Martínek, T.: Prednáška návrh externých adaptérov a vestavených systémov. [Online; navštíveno 28.12.2009].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/NAV-IT/texts/nav2.pdf

[12] Kořenek, J.; Martínek, T.: Prednáška Pokročilých číslicových systémov. [Online; navštíveno 26.12.2009].
URL https://www.fit.vutbr.cz/study/courses/PCS/private/prednasky/02_hdl_jazyky_I/vhdl.pdf

[13] Sekanina, L.: Prednáška návrh počítačovych systémov. [Online; navštíveno 23.12.2009].

URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/INP-IT/texts/inp_vhdl.pdf

[14] Tyson, J.; Grabianowski, E.: How PCI Works. [Online; navštíveno 2.1.2010].
URL http://computer.howstuffworks.com/pci.htm

[15] Wikipedia: PCI — Wikipedia, The Free Encyclopedia. [Online; navštíveno 4.1.2010].
URL http://en.wikipedia.org/wiki/Conventional_PCI

[16] Xilinx: 0 Spartan-II/Spartan-IIE Family OTP Configuration PROMs (XC17S00A). [Online; navštíveno 10.10.2009].
URL http://www.xilinx.com/support/documentation/data_sheets/ds078.pdf

[17] Xilinx: Interface PCI. [Online; navštíveno 5.2.2010].
URL http://www.xilinx.com/products/design_resources/conn_central/protocols/pci_pcix.htm

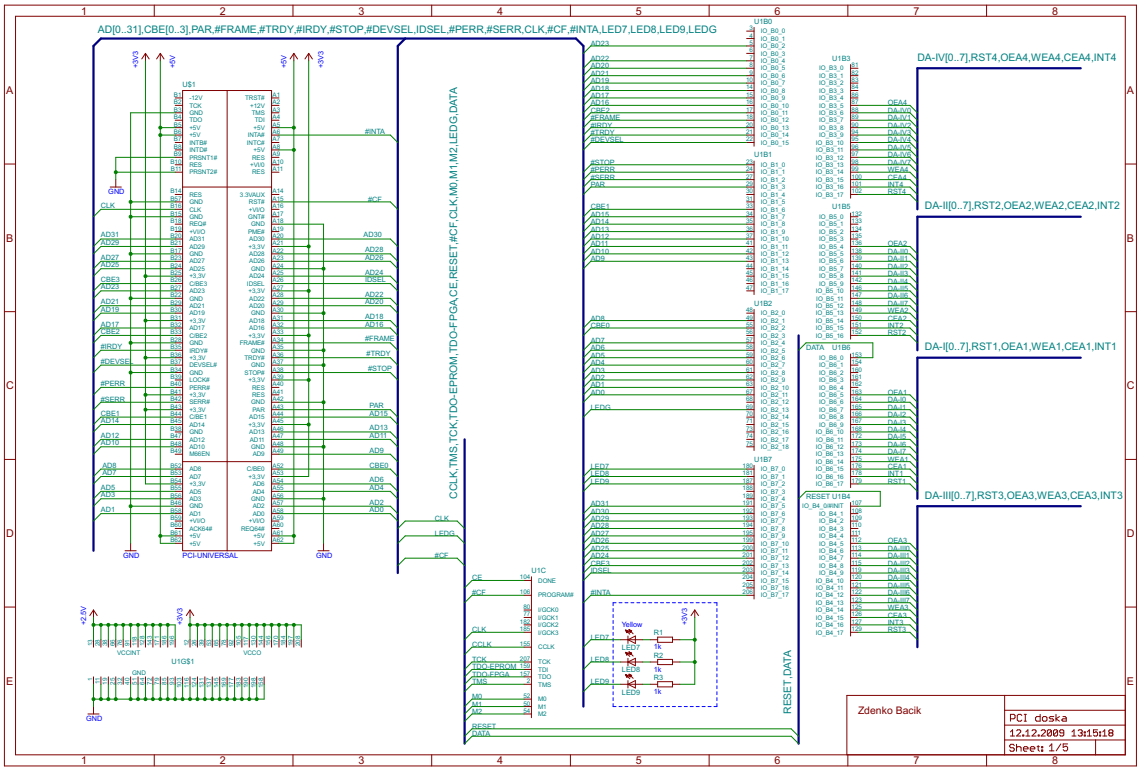
[18] Xilinx: Spartan-II FPGA Family Data Sheet. [Online; navštíveno 22.9.2009].
URL http://www.xilinx.com/support/documentation/data_sheets/ds001.pdf

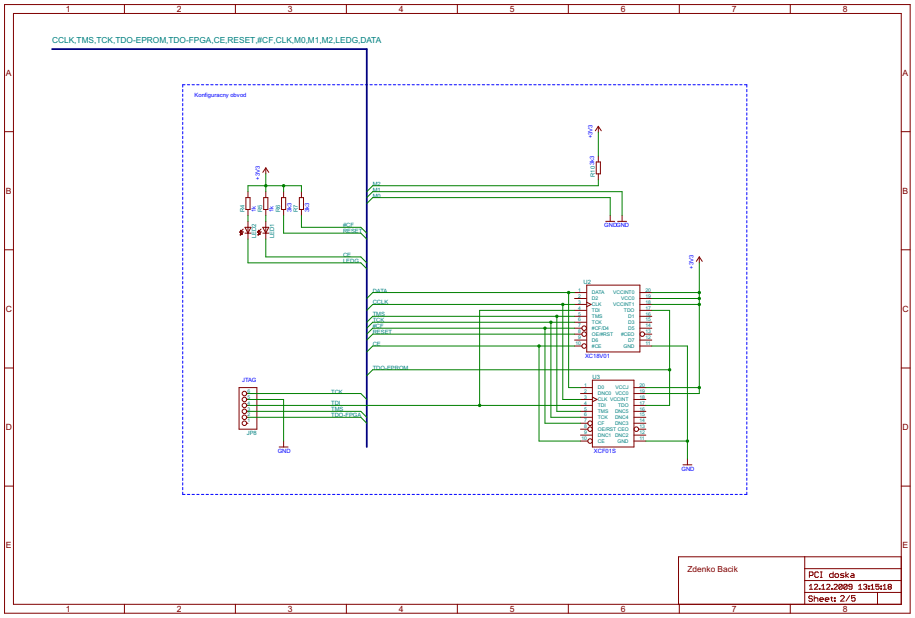
[19] Xilinx: XC18V00 Series In-System-Programmable Configuration PROMs. [Online; navštíveno 10.10.2009].
URL http://www.xilinx.com/support/documentation/data_sheets/ds026.pdf

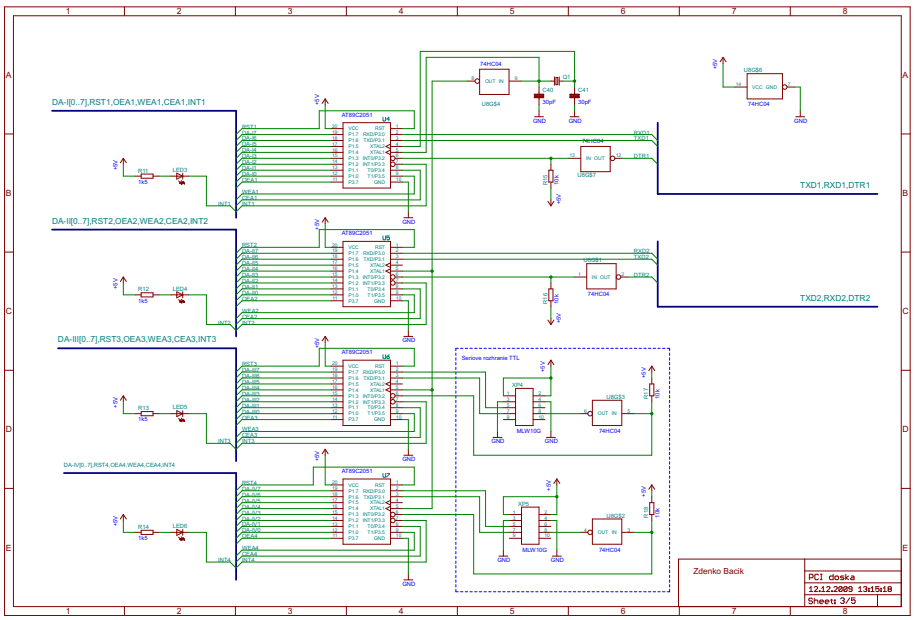
Seznam příloh

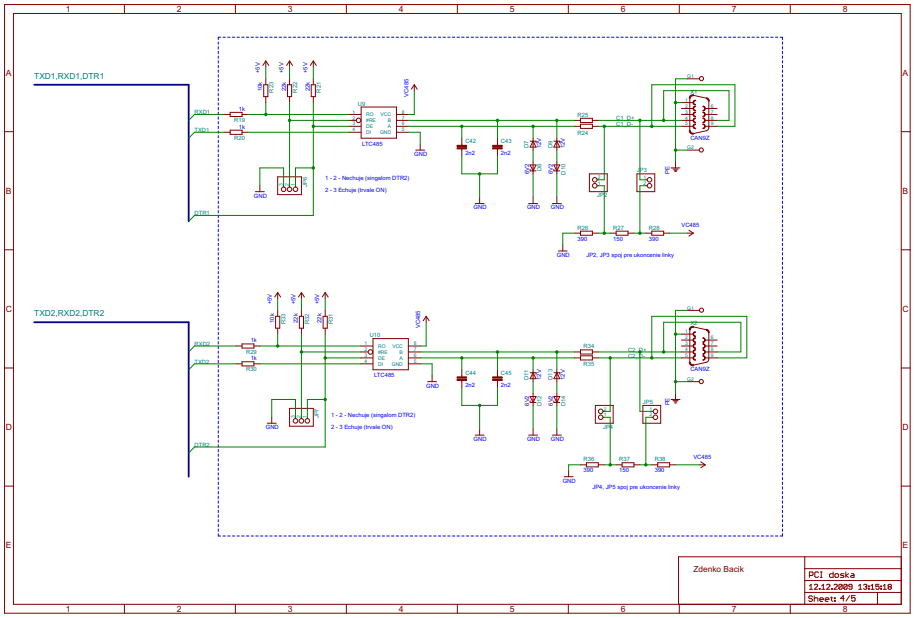
Dodatek A

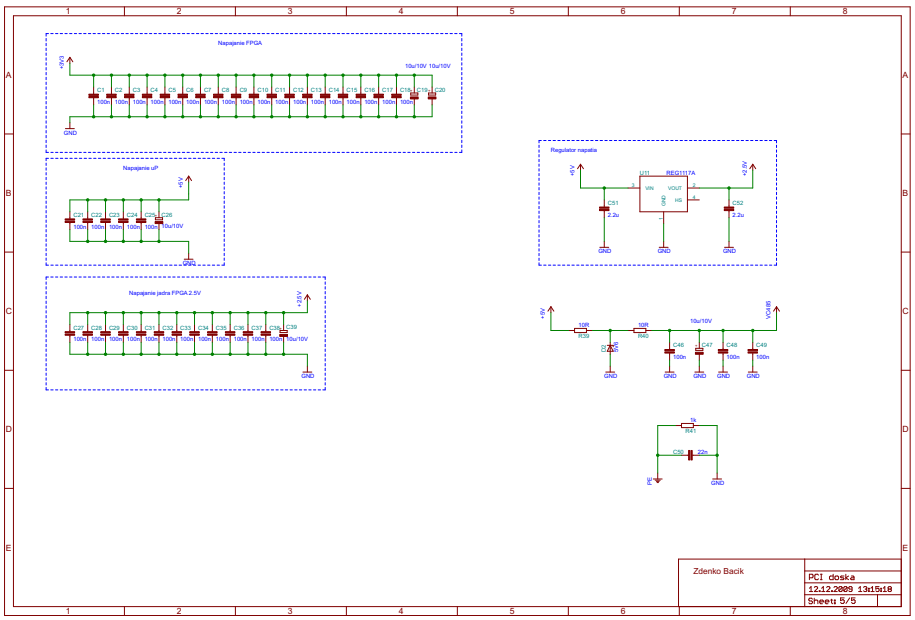
Schéma komunikačnej karty




CCLK,TMS,TCK,TDO-EPROM,TDO-FPGA,CE,RESET,#CF,CLK,M0,M1,M2,LEDG,DATA
Konfiguracny obvod
JTAG
XC18V01
XCF01S
Zdenko Bacik
PCI doska
12.12.2009 13:15:18
Sheet: 2/5

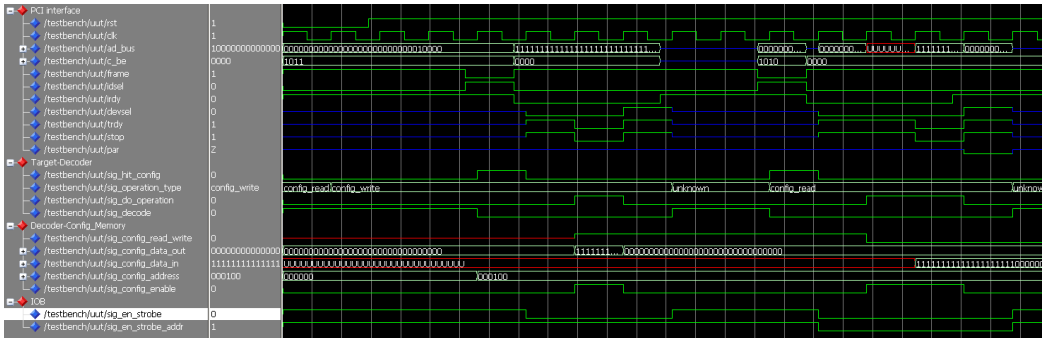

DA-I[0..7],RST1,OEA1,WEA1,CEA1,INT1
DA-II[0..7],RST2,OEA2,WEA2,CEA2,INT2
DA-III[0..7],RST3,OEA3,WEA3,CEA3,INT3
DA-IV[0..7],RST4,OEA4,WEA4,CEA4,INT4
TXD1,RXD1,DTR1
TXD2,RXD2,DTR2
Sietove rozhranie TTL
AT89C2051
74HC04
MLW10G
Zdenko Bacik
PCI doska
12.12.2009 13:15:18
Sheet: 3/5


TXD1,RXD1,DTR1
TXD2,RXD2,DTR2
LTC485
1 - 2 - Nechuje (singalom DTR2)
2 - 3 Echuje (invale ON)
JP2, JP3 spoj pre ukoncenie linky
JP4, JP5 spoj pre ukoncenie linky
CANBZ
Zdenko Bacik
PCI doska
12.12.2009 13:15:18
Sheet: 4/5


Napajanie FPGA
Napajanie uP
Napajanie jadra FPGA 2.5V
Regulator napatia
REG1117A
Zdenko Bacik
PCI doska
12.12.2009 13:15:18
Sheet: 5/5

Dodatek B

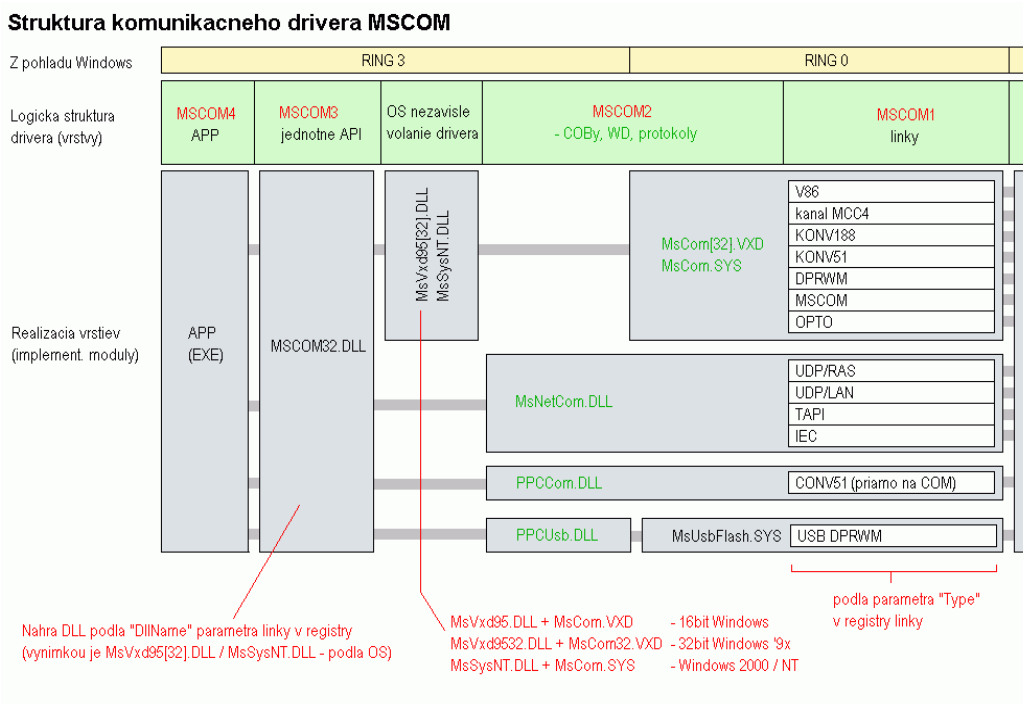
Výsledná simulácia



Obrázek B.1: Výsledná simulácia.

Dodatek C

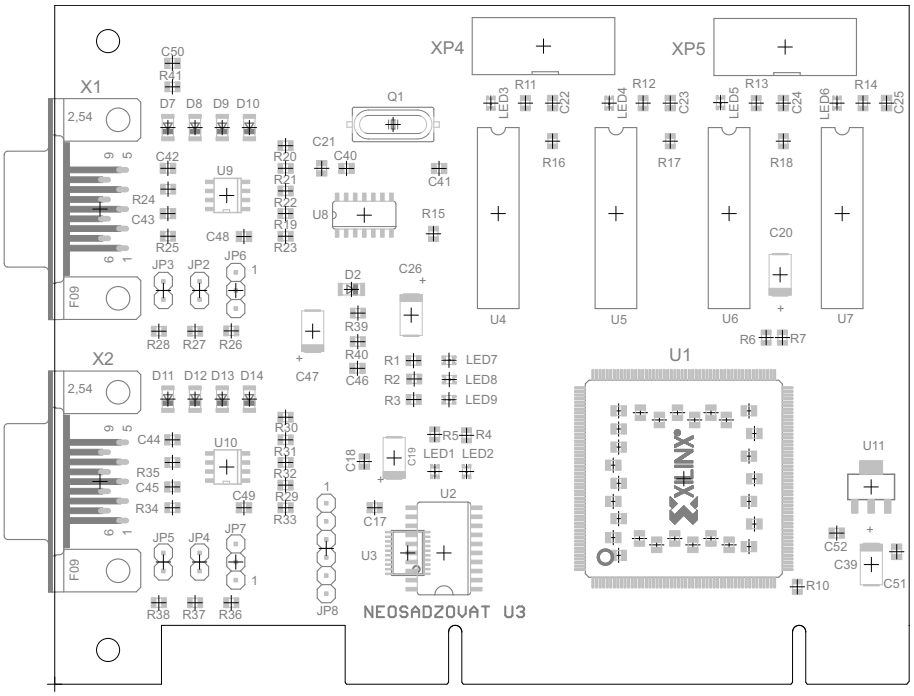
Štruktúra ovládača



Obrázek C.1: Štruktúra ovládača

Dodatek D

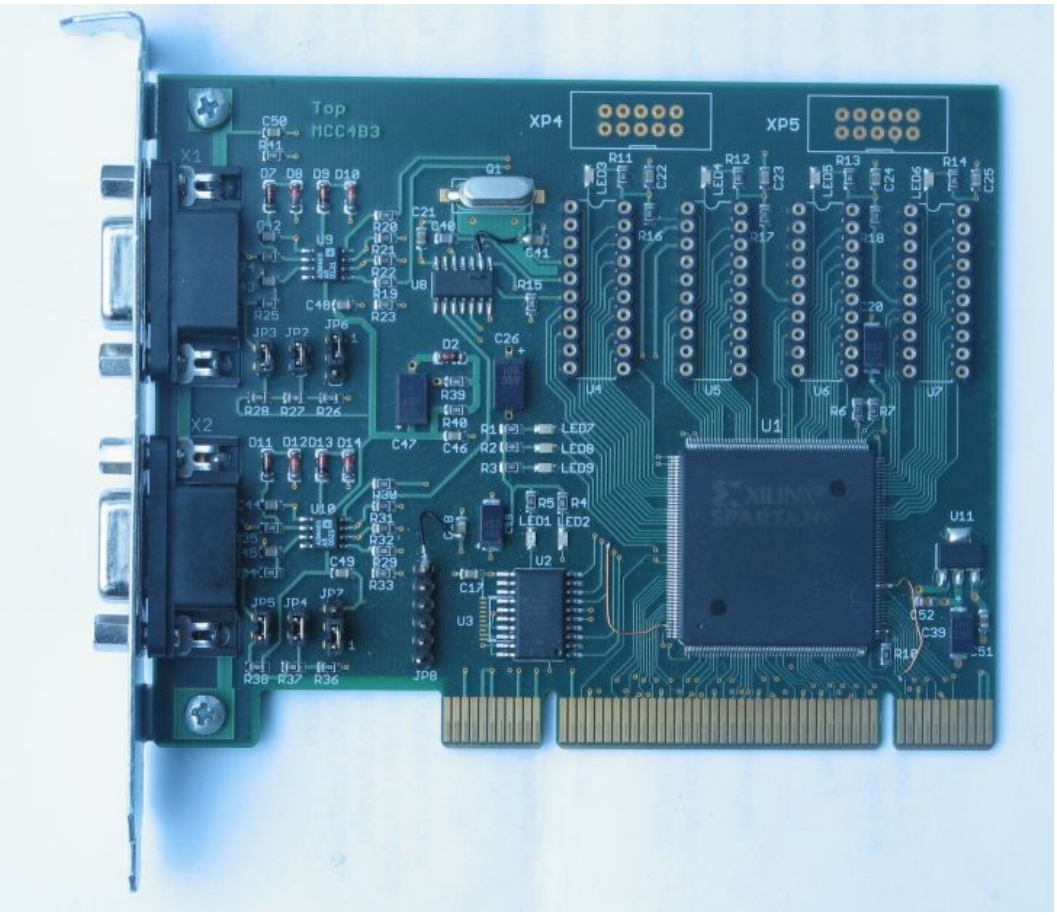
Osadenie dosky plošných spojov



Obrázek D.1: Osadenie plošného spoja súčiastkami.

Dodatek E

Fotodokumentácia



Obrázek E.1: Komunikačná karta PCI.